KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Stochastische Visibiliteit in Rendering Algoritmen a.d.h.v. de occlusion map

Niels Billen

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotor:
Prof. dr. ir. Philip Dutré

Assessoren:
Dr. ir. Thomas Heyman
Dr. ir. Ares Lagae

Begeleider:
Prof. dr. ir. Philip Dutré

Academiejaar 2012 – 2013

Voorwoord

Bij deze wil ik al de mensen bedanken die meegeholpen hebben bij het tot stand brengen van deze thesis. Mijn grootste dank gaat uit naar mijn ouders, die mij niet alleen de kans hebben gegeven om verder te studeren, maar mij ook steeds gesteund hebben gedurende mijn studies.

Christel Geebelen en Bart Nies wil ik heel erg bedanken voor het nalezen en corrigeren van mijn thesis.

Professor Dutré wil ik bedanken voor zijn uitstekende begeleiding gedurende het jaar, zijn bijdragen aan de thesis en de lange en interessante discussies tijdens onze meetings.

Daarnaast wil ik Ares Lagae en professor Dutré nog eens extra bedanken voor de enorme bijdrage die ze geleverd hebben aan het Eurographics paper. Zonder hun ervaring en inbreng zou een acceptatie ondenkbaar geweest zijn.

Ook wil ik Björn Engelen bedanken voor het uitwerken van de theorie rond stochastische visibiliteit in zijn masterproef. Zijn helder geschreven masterproef vormde een stevige basis waarop ik mijn masterproef kon verder bouwen.

Finaal wil ik ook mijn vriendin Katrien bedanken voor al haar steun.

Niels Billen

Inhoudsopgave

Samenvatting

In deze thesis onderzoeken we het concept van stochastische visibiliteit. De zichtbaarheidsbepaling $V(x, y)$ tussen twee punten x en y in een driedimensionale scene is een veel voorkomende, maar kostelijke operatie. De zichtbaarheidsbepaling vereist dat we alle geometrische primitieven tussen de twee punten moeten testen op een intersectie met het lijnsegment $\overline{xy}$.

Met behulp van stochastische visibiliteit splitsen we de visibiliteit $V(x, y)$ in meerdere termen, die elk minder intersectie testen vereisen. We evalueren de visibiliteit stochastisch door voor een schaduwstraal slechts één van deze termen te kiezen en te evalueren. We bewijzen dat deze stochastische visibiliteitsevaluatie zal convergeren naar het juiste resultaat en we zullen aantonen hoe de stochastische visibiliteit kan geïntegreerd worden met een praktisch belichtingsalgoritme.

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

brdf	de bidirectionele reflectie distributie functie
BVH	Bounding Volume Hiërarchie
MSE	Mean Squared Error

Symbolen

$L(x \to \Theta)$	Radiantie van punt x in de richting Θ
$f_r(x, \Theta \leftrightarrow \Psi)$	Bidirectionele reflectie distributie functie voor een inkomende richting Θ en uitgaande richting Ψ in het punt x
$V(x, y)$	de visibiliteitsfunctie tussen twee punten x en y. Deze functie heeft de waarde 1 als x en y onderling zichtbaar zijn. In het andere geval heeft de functie de waarde 0
$V_A(x, y)$	de visibiliteitsfunctie tussen twee punten x en y waarbij we enkel de geometrische primitieven van groep A in rekening brengen
$V_B(x, y)$	de visibiliteitsfunctie tussen twee punten x en y waarbij we enkel de geometrische primitieven van groep B in rekening brengen
$p(x)$	een kansdichtheidsfunctie die aangeeft met welke kans de waarde x voorkomt
$E[x]$	de verwachte waarde van een stochastische variabele x

Hoofdstuk 1

Inleiding

In dit hoofdstuk behandelen we beknopt de noodzakelijke termen en algoritmen van computer graphics waarop we in de volgende hoofdstukken zullen steunen. Hoofdstuk 2 gaat dieper in op de theorie van stochastische visibiliteit. Hoofdstuk 3 introduceert een nieuwe directionele datastructuur, genaamd de occlusion map. In hoofdstuk 4 bespreken we een naïef algoritme dat gebruik maakt van stochastische visibiliteit en de occlusion map. Hoofdstuk 5 breidt het naïve algoritme uit zodat het algoritme een betere performantie heeft. Hoofdstuk 6 bespreekt de eigenschappen van het uitgebreide algoritme.

1.1 Computer graphics

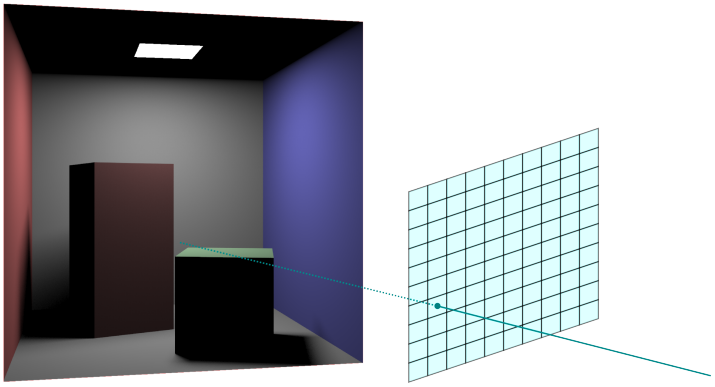
In het domein van computer graphics proberen we een zo realistisch mogelijke afbeelding te genereren vanuit de beschrijving van een driedimensionale wereld. Het vervatten van de wereld in een computer model is reeds een complexe taak. Ten eerste moeten al de objecten worden omgezet in een driedimensionaal computer model. Ten tweede moet de manier waarop licht gereflecteerd wordt door de materialen van de objecten accuraat benaderd worden.

Gegeven de drie dimensionale voorstelling van de wereld, moet een algoritme deze voorstelling omzetten naar een zo realistisch mogelijke afbeelding. Doorheen de jaren zijn hiervoor een aantal algoritmes ontworpen. Ray tracing is nog steeds een zeer populair algoritme voor foto-realistische beeldsynthese omdat ray tracing het fysische transport van licht simuleert. Dit laat toe om zeer complexe lichteffecten te berekenen zoals reflectie en breking van licht door een glazen object. De werking van het ray tracing algoritme bespreken we in de volgende sectie.

1.2 Ray tracing

In deze sectie introduceren we de basis principes van het ray tracing algoritme. Ray tracing simuleert het lichttransport doorheen een driedimensionele scene.

De invoer voor het algoritme is de beschrijving van een driedimensionale scene. Deze beschrijving omvat de positie en oriëntatie van al de computer modellen en



Figuur 1.1: Genereren van zichtstralen. Voor elke pixel zal er een zichtstraal gegenereerd worden. Voor elke zichtstraal zoeken we het eerste intersectie punt in de ruimte. Op dat punt zullen we de belichtingswaarde evalueren.

lichtbronnen, de eigenschappen van al de materialen en een virtuele camera. De virtuele camera beschrijft de positie, richting en kijkhoek waarmee de afbeelding berekend wordt.

Voor iedere pixel van de afbeelding genereren we een zichtstraal vertrekkend uit de camera. Voor deze zichtstraal zoeken we het eerste object in de scene dat deze zichtstraal snijdt. Indien de zichtstraal geen enkel object in de scene snijdt, dan krijgt de pixel een op voorhand gedefinieerde achtergrondkleur (bijvoorbeeld zwart).

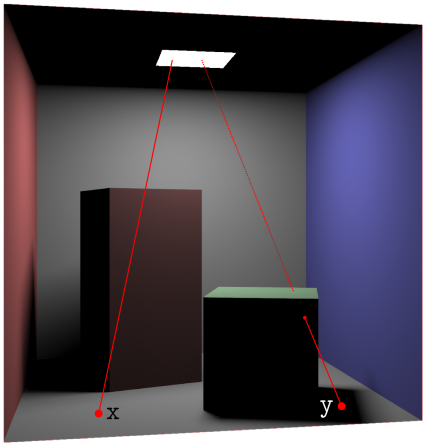
Indien de zichtstraal wel een object snijdt, dan zoeken we het exacte snijpunt x van de zichtstraal met het gesneden object. Vervolgens evalueren we de materiaaleigenschappen van het object in het punt x. Deze evaluatie zal de belichting van het object in het punt x bepalen en zal hierdoor ook de belichtingswaarde van de pixel bepalen. Hoe de evaluatie gebeurt is afhankelijk van het materiaal en kan aanleiding geven tot verdere recursieve stralen die doorheen de scene gevolgd moeten worden.

- **Diffuse materialen**: de belichting in het punt x is enkel afhankelijk van de intensiteit en de positie van de lichtbron. In dit geval moet bepaald worden of het punt x zichtbaar is voor de lichtbron. De visibiliteit wordt geëvalueerd door een schaduwstraal doorheen de scene te volgen, vertrekkend vanuit het punt x in de richting van de lichtbron. Indien de schaduwstraal de lichtbron als eerste snijdt, dan weten we dat het punt x zichtbaar is voor de lichtbron. Het volgen van schaduwstralen wordt geïllustreerd in figuur 1.2.

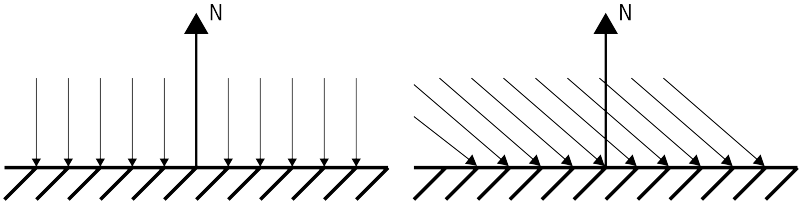
  Indien het punt x belicht wordt, moeten we evalueren hoe het materiaal het inkomende licht van de lichtbron reflecteert. Een voorbeeld van een zeer eenvoudig materiaal model is dat van Lambert:

  $$c = c_l c_r max\,(0, \cos\alpha) \tag{1.1}$$

  Lambert merkte op dat de hoeveelheid licht dat gereflecteerd wordt door een diffuus oppervlak afhankelijk is van de hoek die het invallende licht maakt



FIGUUR 1.2: Volgen van schaduwstralen doorheen de scene. De schaduwstraal vertrekkend vanuit het punt x bereikt de lichtbron. De schaduwstraal vertrekkend vanuit het punt y wordt geblokkeerd door de kleine kubus. Het punt y ligt in de schaduw.



FIGUUR 1.3: Oorsprong van de geometrische term. Stralen die loodrecht op het oppervlak invallen hebben een grotere bijdrage dan stralen die onder een hoek invallen.

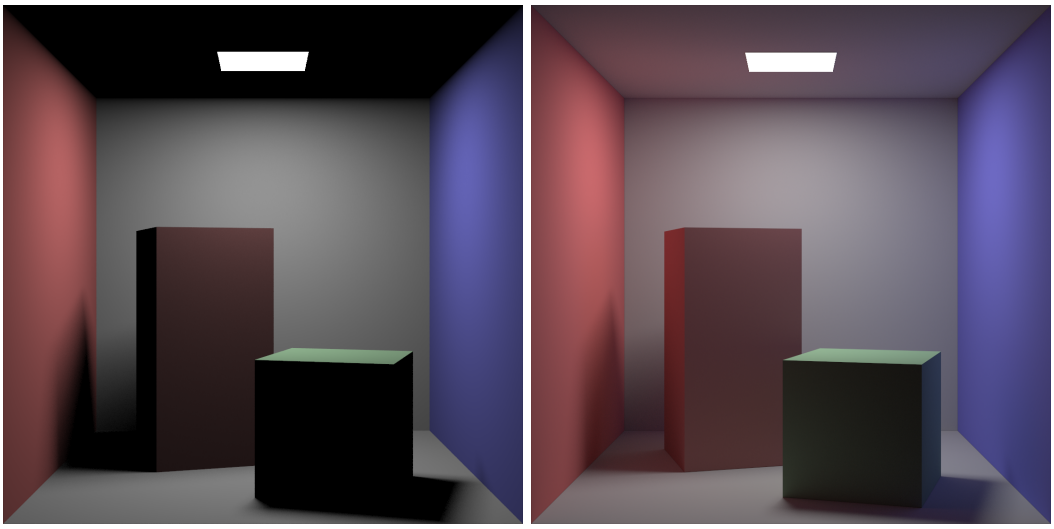
ten opzichte van het oppervlak. Wanneer het licht loodrecht invalt op het oppervlak, dan is de bijdrage groter dan wanneer het licht invalt onder een hoek. Dit wordt afgebeeld in figuur 1.3. De uiteindelijke belichting is dan gelijk aan de lichtintensiteit c_l geschaald met de reflectiecoëfficiënt van het materiaal c_r, geschaald met de cosinus van de hoek tussen de normaal van het oppervlak en de richting van het invallende licht.

- **Speculaire materialen**: materialen zoals spiegels en metalen weerkaatsen licht. In dit geval schieten we een recursieve zichtstraal vanuit het punt x in de spiegelende richting. De belichtingsevaluatie zal dan gebeuren op het eerste intersectie punt van de recursieve zichtstraal.

- **Refractieve materialen**: materialen zoals glas breken het licht. Hierbij reflecteert een deel in een perfect spiegelende richting, terwijl een ander deel wordt doorgelaten door het oppervlak. In dit geval zijn dus twee recursieve zichtstralen nodig: een die de bijdrage van de spiegelende richting berekent en een die de bijdrage in de gebroken richting berekent.

1.3 De rendering vergelijking

Het ray tracing algoritme zoals uitgelegd in sectie 1.2 kan reeds vrij realistische afbeeldingen genereren. Het beschreven algoritme brengt wel enkel de directe belichting en de belichting van spiegelende oppervlakken in rekening. Dit wil zeggen dat het licht dat indirect tussen twee oppervlakken gewisseld wordt, genegeerd wordt. Het is net het indirecte licht dat een afbeelding realistisch maakt (vergelijk de afbeeldingen in figuur 1.4)



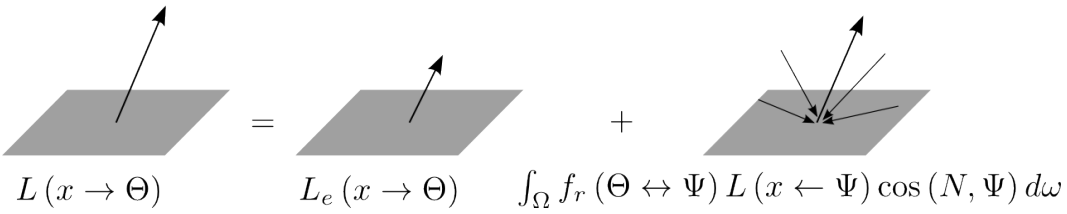
Figuur 1.4: Cornell box links gerenderd met enkel de directe belichting en rechts gerenderd met globale belichting.

De vergelijking die de verschijnselen van lichttransport uitdrukt, is de rendering vergelijking. Deze vergelijking is een recursieve integraal die uitdrukt hoe licht of radiantie zich verplaatst doorheen de ruimte [3]:

$$L\left(x \to \Theta\right) = L_e\left(x \to \Theta\right) + \int_{\Omega} f_r\left(x, \Theta \leftrightarrow \Psi\right) L\left(x \leftarrow \Psi\right) \cos\left(N, \Psi\right) d\omega_{\Psi} \qquad (1.2)$$

De rendering vergelijking stelt dat de hoeveelheid radiantie die vanuit x vertrekt in richting Θ, opgebroken kan worden in een som van twee termen (zie figuur 1.5). De eerste term $L_e\left(x \to \Theta\right)$ stelt de eigen emissie voor van het punt. Deze term is enkel verschillend van nul voor objecten die zelf licht uitstralen. De tweede term berekent hoeveel radiantie er wordt gereflecteerd in het punt x. Dit is de som van al het licht dat invalt vanuit de richtingen Ψ van de hemisfeer Ω boven het punt x. Deze hoeveelheid radiantie $L\left(x \leftarrow \Psi\right)$ wordt geschaald met een geometrische term $\cos\left(N, \Psi\right)$ en met de bidirectionele reflectie distributie functie $f_r\left(x, \Theta \leftrightarrow \Psi\right)$. $d\omega_{\Psi}$ is de infinitesimale ruimtehoek waarin het licht $L\left(x \leftarrow \Psi\right)$ binnenvalt.

De *brdf* of bidirectionele reflectie distributie functie f_r hangt af van het materiaal van een object. De functie geeft aan hoeveel procent van het licht dat invalt vanuit de richting Θ, gereflecteerd wordt in de richting Ψ. Deze functie kan variëren van een



FIGUUR 1.5: De rendering vergelijking kan opgedeeld worden in twee termen. De eerste term is de emissie van een object. De tweede term berekent hoe het invallende licht gereflecteerd wordt.

constante functie voor een materiaal dat in elke richting evenveel licht uitstraalt, tot een functie die meetresultaten van een opgemeten materiaal interpoleert. Men moet wel opletten dat de som van de *brdf* over alle inkomende en uitgaande richtingen steeds kleiner is dan 1. Indien er niet aan deze voorwaarde voldaan is, dan zou dit betekenen dat er meer licht gereflecteerd wordt dan dat er invalt op het materiaal. Daarnaast geldt voor fysisch correcte *brdf*'s dat de hoeveelheid licht dat gereflecteerd wordt van Θ naar Ω gelijk moet zijn aan de hoeveelheid licht dat gereflecteerd wordt van Ω naar Θ.

De cosinus term $\cos(N, \Psi)$ is een geometrische term die afkomstig is van de oriëntatie van het oppervlak in het punt x zoals afgebeeld in figuur 1.3.

1.4 Monte Carlo integratie

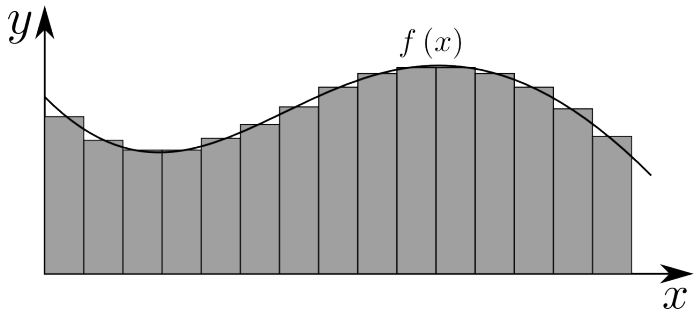
De rendering vergelijking is een multidimensionale integraal. Om eendimensionale integralen op te lossen, gebruikt men gewoonlijk quadratuur methodes. Deze methodes bemonsteren de functie op regelmatige intervallen om de oppervlakte onder de functie te schatten (zie figuur 1.6). Deze methode is echter niet praktisch voor multidimensionale integralen. Indien we elke dimensie bemonsteren met N monsters, dan is het totaal aantal monsters voor een d-dimensionale integraal gelijk aan N^d. Het aantal monsters neemt dus exponentieel toe met de dimensie van de integraal, waardoor quadratuur formules onpraktisch worden.

De techniek die dikwijls gebruikt wordt om multidimensionale integralen op te lossen, is Monte Carlo integratie. Deze techniek heeft het voordeel dat we, ongeacht de dimensie, vrij zijn om het aantal monsters N te kiezen.

De N monsters worden willekeurig in het integratiedomein gekozen. De gemiddelde waarde van deze monsters convergeert naar de exacte waarde van de integraal. Stel dat we de integraal willen berekenen van de functie $f(x)$ over het domein $[a, b]$:

$$I = \int_a^b f(x)\, dx \tag{1.3}$$

dan is $\langle I \rangle$ een schatter voor de integraal I die convergeert naarmate het aantal



Figuur 1.6: Eendimensionale integratie met behulp van quadratuur formules.

monsters N toeneemt:

$$I \approx \langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{1.4}$$

Hierbij is x_i een willekeurig monster uit het integratiedomein $[a, b]$ en $p(x_i)$ de kansdichtheid waarmee we dit monster kozen. Wanneer de monsters volledig willekeurig gekozen worden, dan is deze kansdichtheid voor elk monster gelijk aan $1/(b-a)$ waardoor de formule vereenvoudigt tot:

$$I \approx \langle I \rangle = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i) \tag{1.5}$$

De uitbreiding naar multidimensionale integralen is triviaal:

$$I = \int \int f(x, y)\, dxdy \tag{1.6}$$

$$\langle I \rangle = \frac{1}{N} \sum_{i}^{N} \frac{f(x_i, y_i)}{p(x_i, y_i)} \tag{1.7}$$

Bij Monte Carlo simulaties zijn we vaak geïnteresseerd in de variantie of de gemiddelde afwijking van de schatter ten opzichte van het resultaat. De variantie van een Monte Carlo schatter kan berekend worden met de volgende formule:

$$\sigma^2[\langle I \rangle] = \frac{1}{N} \int \left(\frac{f(x)}{p(x)} - I \right)^2 p(x)\, dx \tag{1.8}$$

Aan deze formule zien we dat de variantie omgekeerd evenredig is met het aantal monsters N.

In het volgende voorbeeld zullen we het gebruik van Monte Carlo integratie illustreren door het getal π te benaderen. Hiervoor moeten we de volgende integraal oplossen:

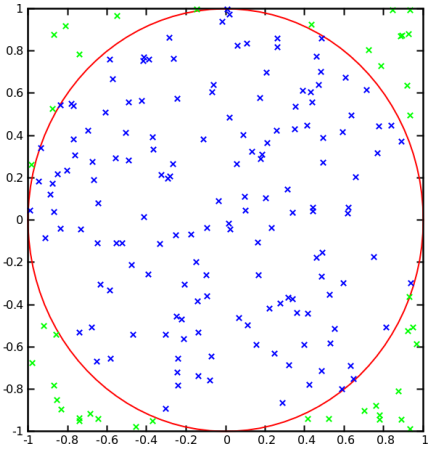
$$\pi = \int_{-1}^{1} \int_{-1}^{1} f(x, y) \tag{1.9}$$

$$f(x, y) = \begin{cases} 1 & x^2 + y^2 \leq 1 \\ 0 & x^2 + y^2 > 1 \end{cases} \tag{1.10}$$

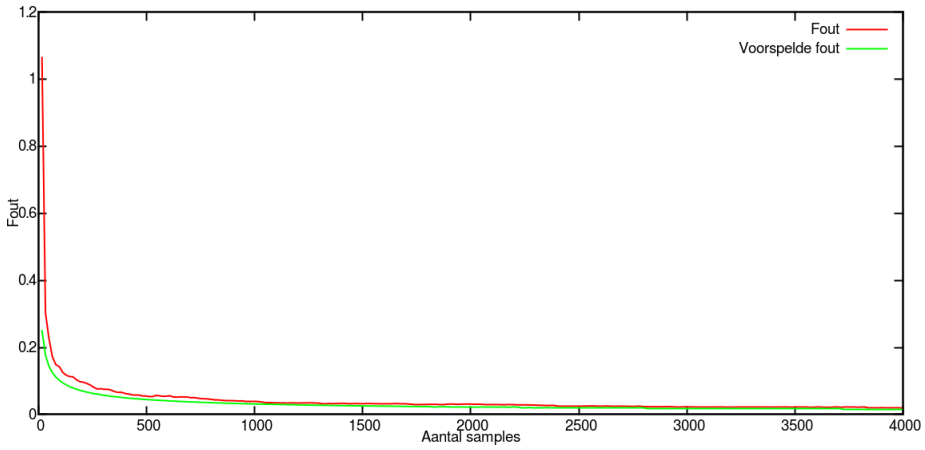
Voor de Monte Carlo integratie van deze integraal genereren we willekeurige monsters in het integratiedomein $[-1,1]^2$. De kansdichtheid $p(x_i, y_i)$ voor elk van deze monsters is gelijk aan $1/4$. De schatter voor de waarde van π is dan gelijk aan:

$$\langle I \rangle = \frac{4}{N} \sum_{i=1}^{N} f(x_i, y_i) \,. \tag{1.11}$$

Figuur 1.7 toont de eerste 200 monsters voor de Monte Carlo simulatie. Van deze 200 monsters zijn er 158 die binnen de eenheidscirkel liggen. De schatter heeft dus een waarde gelijk aan 3.16. De afwijking ten opzichte van de echte waarde van π is dus al kleiner dan 1%. Figuur 1.8 toont dat de fout σ van de schatter evenredig is met $\frac{1}{\sqrt{N}}$.



FIGUUR 1.7: De eerste 200 monsters van de Monte Carlo simulatie. Van de 200 monsters vallen er 158 monsters binnen de cirkel waardoor de schatter gelijk is aan 3.16. Na 200 monsters is de afwijking dus reeds kleiner dan 1%.



FIGUUR 1.8: Fout van de Monte Carlo schatter ten opzichte van de geschatte fout. De geschatte fout σ is hierbij evenredig met $1/\sqrt{N}$.

1.4.1 Variantie reductie

Zoals vermeld in de vorige sectie is de standaarddeviatie of de fout van de Monte Carlo schatter omgekeerd evenredig met de vierkantswortel van het aantal monsters. We kunnen de convergentie verbeteren door de monsters beter te verdelen over het integratiedomein. We vermelden hier de belangrijkste variantie reductietechnieken.

Stratificatie

Wanneer we de monsters willekeurig kiezen, gebeurt het vaak dat deze niet evenredig verspreid zijn over het hele integratiedomein. Om dit samenklonteren van monsters tegen te gaan, verdelen we het integratiedomein in een aantal disjuncte strata. Binnen elke stratum genereren we dan eenzelfde aantal monsters.

Over het algemeen reduceert stratificatie de variantie. We kunnen wiskundig bewijzen dat de variantie met stratificatie nooit groter is dan de variantie met volledig willekeurige monsters [6].

Figuur 1.10 toont een reeks van 128 gestratificeerde monsters in hun overeenkomstige strata. Hoewel de monsters nu evenrediger zijn verspreid over het integratiedomein, zien we bijvoorbeeld rechtsboven dat het samenklonteren van monsters nog steeds kan voorkomen.

Quasi-Monte Carlo

Quasi-Monte Carlo technieken gebruiken volledig deterministische sequenties om hun monsters te genereren. Deze sequenties gedragen zich echter willekeurig en de opeenvolgende monsters hebben als eigenschap dat ze evenredig verdeeld zijn over het hele integratiedomein. Hierdoor ligt de convergentiesnelheid van Quasi-Monte Carlo hoger dan standaard Monte Carlo. De fout op de schatter daalt nu evenredig met $(\log N)^d / N$ wat voor grote N en lage dimensies d veel beter is dan $1/\sqrt{N}$.

Figuur 1.11 toont een quasi willekeurige sequentie. In vergelijking met de willekeurige en gestratificeerde sequentie zien we dat de monsters zeer gelijkmatig verdeeld zijn over het integratiedomein.

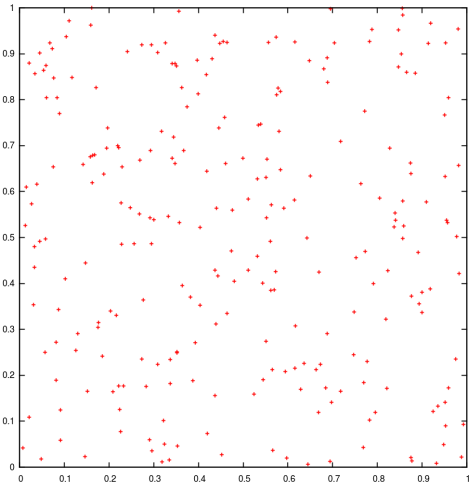
Importance sampling

Een andere techniek om de variantie te reduceren is de monsters niet uniform over het integratiedomein te verspreiden. Hierbij gaan we de monsters dus kiezen aan de hand van een kansdichtheidsfunctie $p(x_i)$ waarbij de kansen niet uniform zijn.

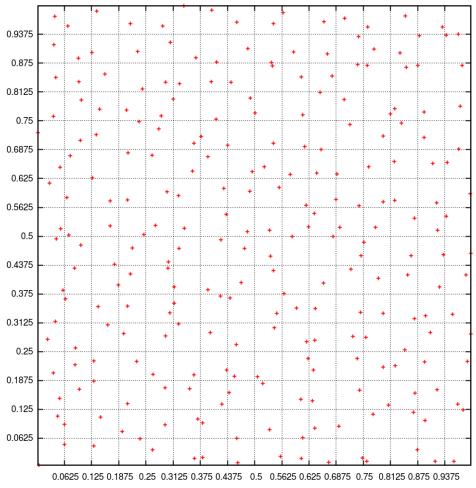
We kunnen bewijzen dat voor een functie $f(x)$ die positief is over het hele integratiedomein D, de optimale kansdichtheid gelijk is aan:

$$p_{optimaal}(x) = \frac{|f(x)|}{\int_D f(x)} \tag{1.12}$$

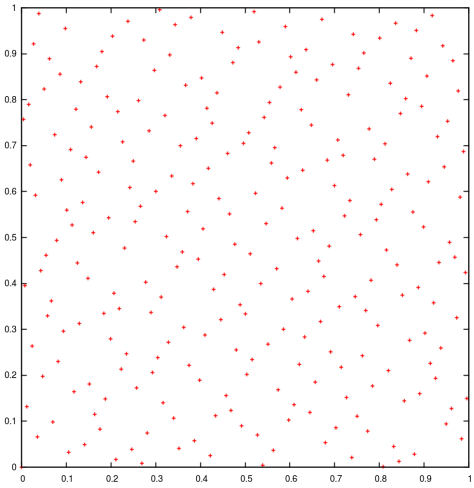
Hiervoor hebben we echter kennis nodig van de exacte integraal. Deze kennis hebben we niet op voorhand, maar het volstaat echter om een kansdichtheidsfunctie te nemen die het verloop van de functie $f(x)$ volgt.



FIGUUR 1.9: Willekeurige sequentie.



FIGUUR 1.10: Gestratificeerde sequentie.



FIGUUR 1.11: Quasi-Monte Carlo sequentie.

1.5 Path tracing

Monte Carlo integratie kan ook worden gebruikt om de rendering vergelijking op te lossen. Herinner dat de rendering vergelijking gelijk is aan:

$$L(x \to \Theta) = L_e(x \to \Theta) + L_r(x \to \Theta) \tag{1.13}$$

$$= L_e(x \to \Theta) + \int_\Omega f_r(x, \Theta \leftrightarrow \Psi) L(x \leftarrow \Psi) \cos(N, \Psi) \, d\omega_\Psi \tag{1.14}$$

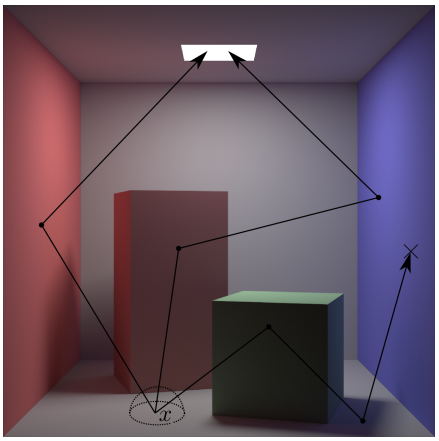
Om de integraal $L_r(x \to \Theta)$ te berekenen gebruiken we de volgende Monte Carlo schatter:

$$\langle L_r(x \to \Theta) \rangle = \frac{1}{N} \sum_i^N \frac{L(x \leftarrow \Psi_i) f_r(x, \Theta \leftrightarrow \Psi_i) \cos(\Psi_i, N)}{p(\Psi_i)} \tag{1.15}$$

Elke term in de schatter is triviaal om te evalueren buiten de term $L(x \leftarrow \Psi_i)$. Een van de eigenschappen van radiantie in vacuüm is echter dat:

$$L(x \leftarrow \Psi_i) = L(y \to -\Psi_i) \tag{1.16}$$

waarbij y het eerste intersectie punt is in de scene van een straal vertrekkend uit x in de richting Θ. Hierdoor kunnen we de term $L(x \leftarrow \Psi_i)$ zelf schatten met vergelijking 1.15. De vergelijking is dus recursief en eindigt enkel wanneer we een lichtbron raken. Het path tracing algoritme bouwt dus recursief een pad op vanuit de camera naar de lichtbron. Te lange paden worden afgebroken met behulp van Russische roulette. Dit is een stochastische methode die een pad afbreekt met een bepaalde kans $1 - p$. Ter compensatie moet elke bijdrage geschaald worden met $1/p$. Figuur 1.12 toont hoe het path tracing algoritme paden construeert.



Figuur 1.12: Illustratie van enkele gegenereerde paden door het path tracing algoritme. De paden die de lichtbron bereiken zullen bijdragen tot de belichtingsevaluatie. De paden die met behulp van Russische roulette worden afgebroken zullen niet bijdragen tot de finale belichting.

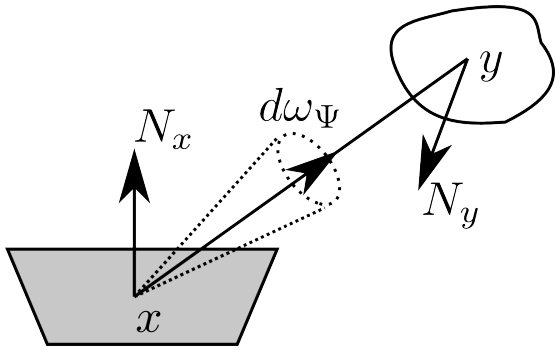
1.6 Visibility sampling

De kern van deze thesis is de visibiliteitsfunctie $V(x, y)$. We zagen reeds in sectie 1.2 dat deze voorkomt wanneer we een schaduwstraal volgen naar de lichtbron. Deze operatie verschijnt in de rendering vergelijking uit sectie 1.3 als we de integraal over de hemisfeer herschrijven als een integraal over de oppervlakken van de scene. De rendering vergelijking over de oppervlakken van de scene wordt dikwijls gebruikt om zeer efficiënt de directe belichting te berekenen. Voor de directe belichting hoeven we immers enkel te integreren over de oppervlakken van de lichtbronnen.

Om de hemisfeer integraal om te zetten naar een oppervlakte integraal, herschrijven we de infinitesimale ruimtehoek $d\omega_\Psi$ als volgt naar een infinitesimaal oppervlak dA_y.

$$d\omega_\Psi = d\omega_{x \leftarrow dA_y} = \cos(N_y, -\Psi) \frac{dA_y}{r_{xy}^2} \tag{1.17}$$

De ruimtehoek $d\omega_\Psi$ is gelijk aan de ruimtehoek van een infinitesimale oppervlak dA_y dat zichtbaar is vanuit x. De ruimtehoek van een oppervlak is de hoek over de hemisfeer rond het punt x opgespannen door dat oppervlak. Deze ruimtehoek is gedefinieerd als $\cos(N_y, -\Psi) \frac{dA_y}{r_{xy}^2}$, waarbij de cosinus term weer een geometrische term is die compenseert voor de oriëntatie van het oppervlak dA_y. Deze omzetting wordt grafisch geïllustreerd in figuur 1.13.



FIGUUR 1.13: Omzetting van de infinitesimale ruimtehoek $d\omega_\Psi$ naar een infinitesimaal oppervlak dA_y.

Indien we 1.17 invullen in de rendering vergelijking 1.2 dan krijgen we:

$$\begin{aligned} L(x \to \Theta) &= L_e(x \to \Theta) \\ &\quad + \int_A f_r(x, \overrightarrow{xy} \leftrightarrow \Theta) L(y \to \overrightarrow{yx}) V(x, y) G(x, y) dA_y \end{aligned} \tag{1.18}$$

Hierbij staat de term $G(x, y)$ voor de geometrische koppelingsterm:

$$G(x, y) = \frac{\cos(N_x, \Psi) \cos(N_y, -\Psi)}{r_{xy}^2} \tag{1.19}$$

Zoals hierboven vermeld, laat deze vergelijking toe om zeer efficiënt de directe belichting te berekenen. Uit vergelijking 1.14 weten we immers dat de term $L(y \to -\Psi)$ vereenvoudigt tot $L_e(y \to -\Psi)$ als we enkel de directe belichting willen berekenen. Bovendien weten we dat deze term $L_e(y \to -\Psi)$ enkel verschillend van nul zal zijn op de lichtbronnen in de scene.

De rendering vergelijking voor de directe belichting is dus:

$$\begin{aligned} L_{direct}(x \to \Theta) &= L_e(x \to \Theta) \\ &+ \int_{y \in light} f_r(x, \overrightarrow{xy} \leftrightarrow \Theta)\, L_e(y \to \overrightarrow{yx})\, V(x,y)\, G(x,y)\, dA_y \end{aligned} \tag{1.20}$$

De Monte Carlo schatter voor vergelijking 1.20 is:

$$\langle L_{direct}(x \to \Theta) \rangle = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{L_e(y_i \to \overrightarrow{y_i x})\, f_r(x, \Theta \leftrightarrow \overrightarrow{xy_i})\, G(x, y_i)\, V(x, y_i)}{p(y_i)} \tag{1.21}$$

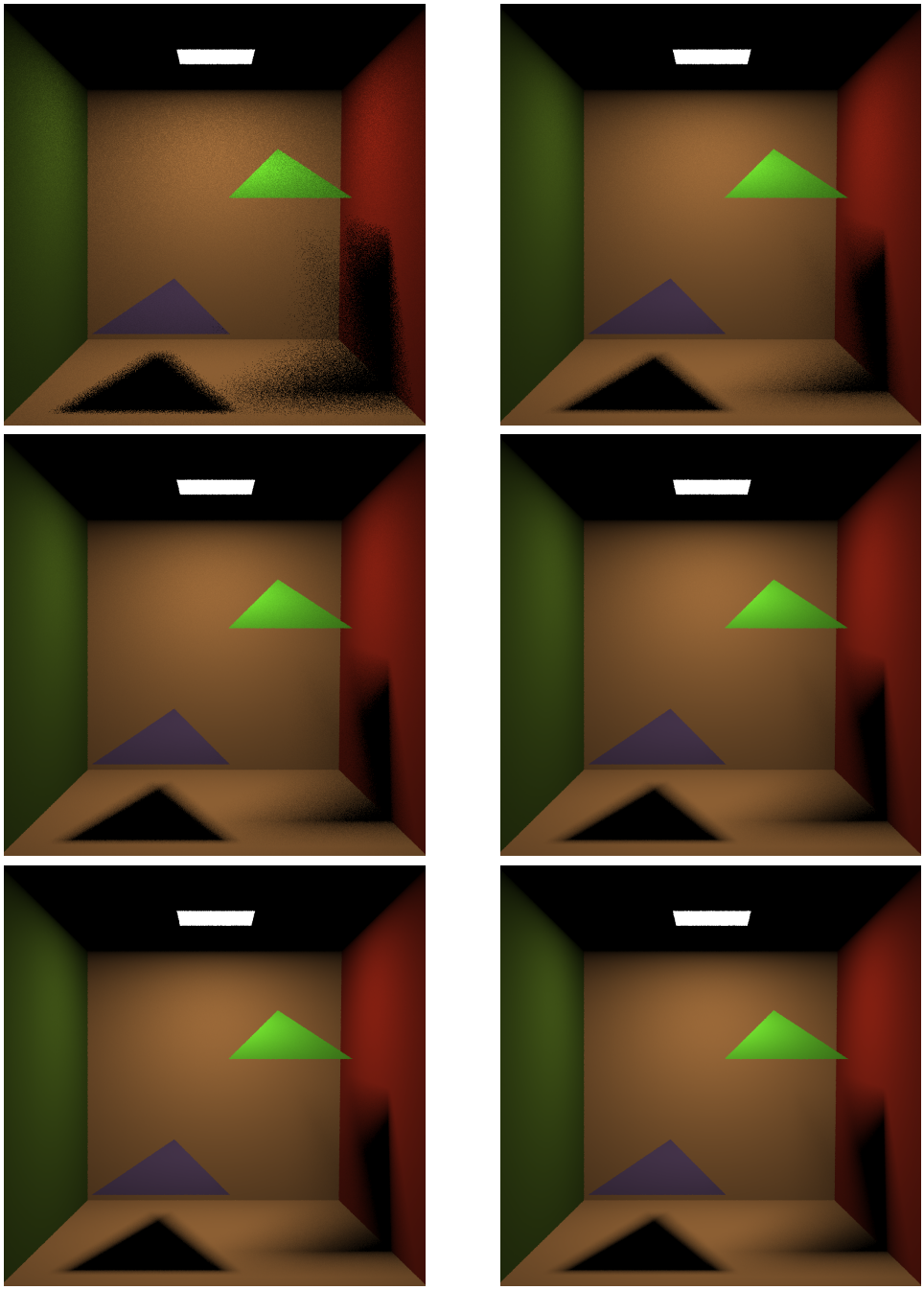
De punten y_i worden volgens de kansdichtheid $p(y_i)$ gegenereerd op de lichtbronnen. Hierbij staat N_s voor het aantal schaduw stralen dat we naar de lichtbronnen sturen. Figuur 1.14 toont hoe deze schatter convergeert voor een stijgend aantal schaduwstralen.

We zien hierbij meteen dat de visibiliteitsoperatie een dure en veel voorkomende operatie is. Voor een afbeelding met P pixels is het aantal visibiliteitsoperaties gelijk aan $P \cdot N_s$. Indien er geen acceleratiestructuur gebruikt wordt, moeten we voor elke visibiliteitsoperatie elk primitief in de scene testen. In een scene met N primitieven is het aantal intersectie testen dus gelijk aan $P \cdot N_s \cdot N$.

1.7 Doelstelling van de thesis

De doelstelling van deze thesis is het versnellen van de visibiliteitsoperatie $V(x,y)$. Zoals besproken in sectie 1.6 komt deze operatie vaak terug in de berekening van de belichting. Om de waarde van de visibiliteit te vinden moeten we elk primitief in de scene testen op een intersectie met het lijnsegment $\bar{xy}$. Met stochastische visibiliteit willen we vermijden dat we exhaustief al de primitieven moeten testen.

Om dit doel te bereiken zullen we de visibiliteit opsplitsen in meerdere termen. Wanneer we een visibiliteitsoperatie moeten uitvoeren, zullen we stochastisch slechts één van deze termen kiezen om te evalueren. Gemiddeld zal het evalueren van één van deze termen minder intersectietesten vergen, maar zal de ruis in de afbeelding door de stochastische evaluatie stijgen.



FIGUUR 1.14: Het bemonsteren van de lichtbron. Deze simpele Cornell box is gerenderd met $N_s = 1, 8, 16, 64, 128, 256$. We zien dat een punt ofwel belicht ofwel in schaduw ligt wanneer we slechts één schaduwstraal gebruiken om de visibiliteit te evalueren. Voor naburige pixels kan dit zeer sterk variëren, waardoor we voor een laag aantal schaduwstralen zien dat er veel ruis aanwezig is. Wanneer we meerdere schaduwstralen gebruiken, zien we dat de schatting verbetert en dat de zachte schaduwregio minder ruis bevat.

Hoofdstuk 2

Theoretisch raamwerk

In dit hoofdstuk bespreken we hoe we de visibiliteit stochastisch kunnen schatten. De eerste sectie verklaart hoe we de visibiliteit stochastisch gaan schatten voor een simpele scene met twee primitieven die schaduw kunnen werpen [4]. In de tweede sectie veralgemenen we het gebruik van stochastische visibiliteit naar scenes met een willekeurig aantal primitieven. Vervolgens onderzoeken we enkele alternatieve decomposities voor de visibiliteit die een snellere convergentie hebben.

2.1 Stochastische visibiliteitsevaluatie

De visibiliteitsfunctie $V(x, y)$ in rendering algoritmen wordt altijd geëvalueerd tussen twee punten x en y. De visibiliteit is gelijk aan 1 wanneer de twee punten onderling zichtbaar zijn. Dit is het geval wanneer geen enkel geometrisch primitief in de scene het lijnsegment $\bar{xy}$ snijdt. Indien er echter één of meer primitieven het lijnsegment $\bar{xy}$ snijden, dan evalueert de visibiliteit tot 0.

Wanneer $Z = \{z_1, z_2, ..., z_n\}$ de verzameling van geometrische primitieven voorstelt die het lijnsegment $\bar{xy}$ kunnen snijden, dan kan de visibiliteitsfunctie geschreven worden als het product van de visibiliteiten van elk geometrisch primitief:

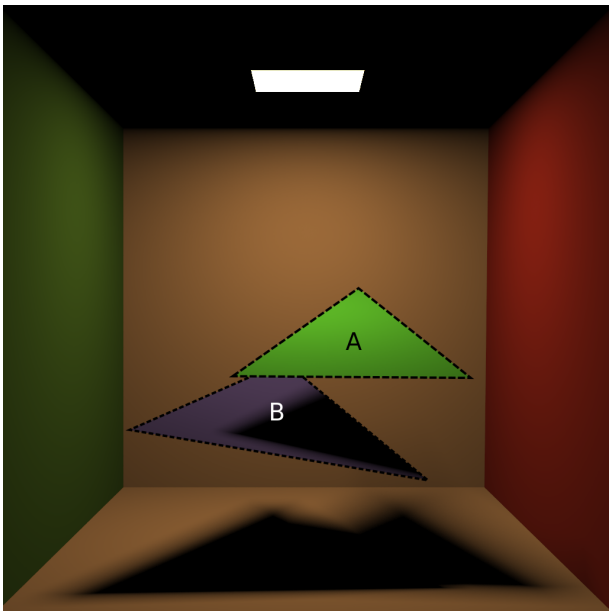
$$V(x, y) = V_{z_1}(x, y) \cdot V_{z_2}(x, y) \cdot ... \cdot V_{z_n}(x, y) = \prod_{i=1}^{n} V_{z_i}(x, y) \tag{2.1}$$

Hierbij evalueert de visibiliteit $V_{z_i}(x, y)$ tot 0 wanneer het geometrisch primitief z_i het lijnsegment $\bar{xy}$ snijdt. Anders is $V_{z_i}(x, y)$ gelijk aan 1.

Om het idee achter de stochastische visibiliteitsevaluatie duidelijker te maken, beschouwen we eerst een scene die slechts twee schaduwwerpende geometrische primitieven bevat. Voor de eenvoud zullen we geometrische primitieven die mogelijk schaduw kunnen werpen voortaan kandidaat blokkers noemen. De scene met twee kandidaat blokkers is afgebeeld in figuur 2.1. Voor deze scene is de visibiliteit gelijk aan:

$$V(x, y) = V_A(x, y)\, V_B(x, y) \tag{2.2}$$

Om deze visibiliteit stochastisch te kunnen schatten, vormen we het visibiliteitsproduct om naar een som. De waarde van een som kan immers stochastisch geschat



Figuur 2.1: Scene met twee kandidaat blokkers A en B.

worden door één van de termen van de som te kiezen en deze term te delen door de kans dat we die term kozen. De som $S = s_1 + s_2 + s_3 + ... + s_n$ kan met andere woorden geschat worden als $\bar{S} = s_i/p_i$.

Om de visibiliteit $V(x, y)$ te ontbinden, gebruiken we de volgende algebraïsche gelijkheid:

$$(1 - a)(1 - b) = 1 - a - b + ab \tag{2.3}$$

$$ab = a + b + (1 - a)(1 - b) - 1 \tag{2.4}$$

Op deze manier kunnen we de visibiliteit schrijven als:

$$V(x, y) = V_A(x, y) V_B(x, y) \tag{2.5}$$

$$= V_A(x, y) + V_B(x, y) + \overline{V_A(x, y)} \cdot \overline{V_B(x, y)} - 1 \tag{2.6}$$

De visibiliteit kan nu geschat worden door één van de volgende drie termen te kiezen met respectievelijk kans p_1, p_2 en p_3:

$$\tilde{V}(x, y) = \begin{cases} \frac{V_A(x,y)}{p_1} & \text{met kans } p_1 \\ \frac{V_B(x,y)}{p_2} & \text{met kans } p_2 \\ \frac{V_A(x,y) \cdot \overline{V_B(x,y)} - 1}{p_3} & \text{met kans } p_3 \end{cases} \tag{2.7}$$

Wanneer we de eerste term kiezen, testen we enkel primitief A. Wanneer we de tweede term kiezen, testen we enkel primitief B. Voor de derde term moeten we zowel primitief A en B testen. Tabel 2.1 toont de verschillende waardes voor iedere term indien we de kansen p_1, p_2 en p_3 gelijkstellen aan $1/3$.

Exacte waarden			**Stochastische evaluatie**		
V_A	V_B	$V_A \cdot V_B$	$3V_A$	$3V_B$	$3\left(\overline{V_A} \cdot \overline{V_B} - 1\right)$
0	0	0	0	0	0
0	1	0	0	3	-3
1	0	0	3	0	-3
1	1	1	3	3	-3

TABEL 2.1: Stochastische evaluatie van de visibiliteit. De drie linkse kolommen geven de waarde van de exacte visibiliteit voor elk van de vier mogelijke visibiliteits-combinaties van twee kandidaat blokkers. De drie rechtse kolommen geven de waarde wanneer we slechts een term kiezen.

Het is zeer eenvoudig om te bewijzen dat de stochastische evaluatie zal convergeren naar de exacte waarde. De verwachte waarde van $\tilde{V}(x,y)$ komt immers overeen met de exacte visibiliteit $V(x,y)$:

$$E[\tilde{V}(x,y)] = p_1 \frac{V_A(x,y)}{p_1} + p_2 \frac{V_B(x,y)}{p_2} + p_3 \frac{\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - 1}{p_3} \tag{2.8}$$

$$= V_A(x,y) + V_B(x,y) + \overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - 1 \tag{2.9}$$

$$= V_A(x,y) \cdot V_B(x,y) \tag{2.10}$$

Het is echter niet intuïtief dat de visibiliteit nu de waardes -3, 0 en 3 kan aannemen. Dit is daarentegen geheel volgens de verwachting vermits we de visibiliteit nu interpreteren als een numerieke waarde en niet meer als een geometrische eigenschap. Dit kunnen we zien wanneer we de stochastische visibiliteit integreren in de rendering vergelijking voor de directe belichting:

$$L(x \to \Theta) = L_e(x \to \Theta) \tag{2.11}$$

$$+ \int_A f_r(\Theta \leftrightarrow y\bar{x})\, L(x \leftarrow y\bar{x})\, G(x,y)\, V_A(x,y) \tag{2.12}$$

$$+ \int_A f_r(\Theta \leftrightarrow y\bar{x})\, L(x \leftarrow y\bar{x})\, G(x,y)\, V_B(x,y) \tag{2.13}$$

$$+ \int_A f_r(\Theta \leftrightarrow y\bar{x})\, L(x \leftarrow y\bar{x})\, G(x,y) \left(\overline{V_A}(x,y)\, \overline{V_B}(x,y) - 1\right) \tag{2.14}$$

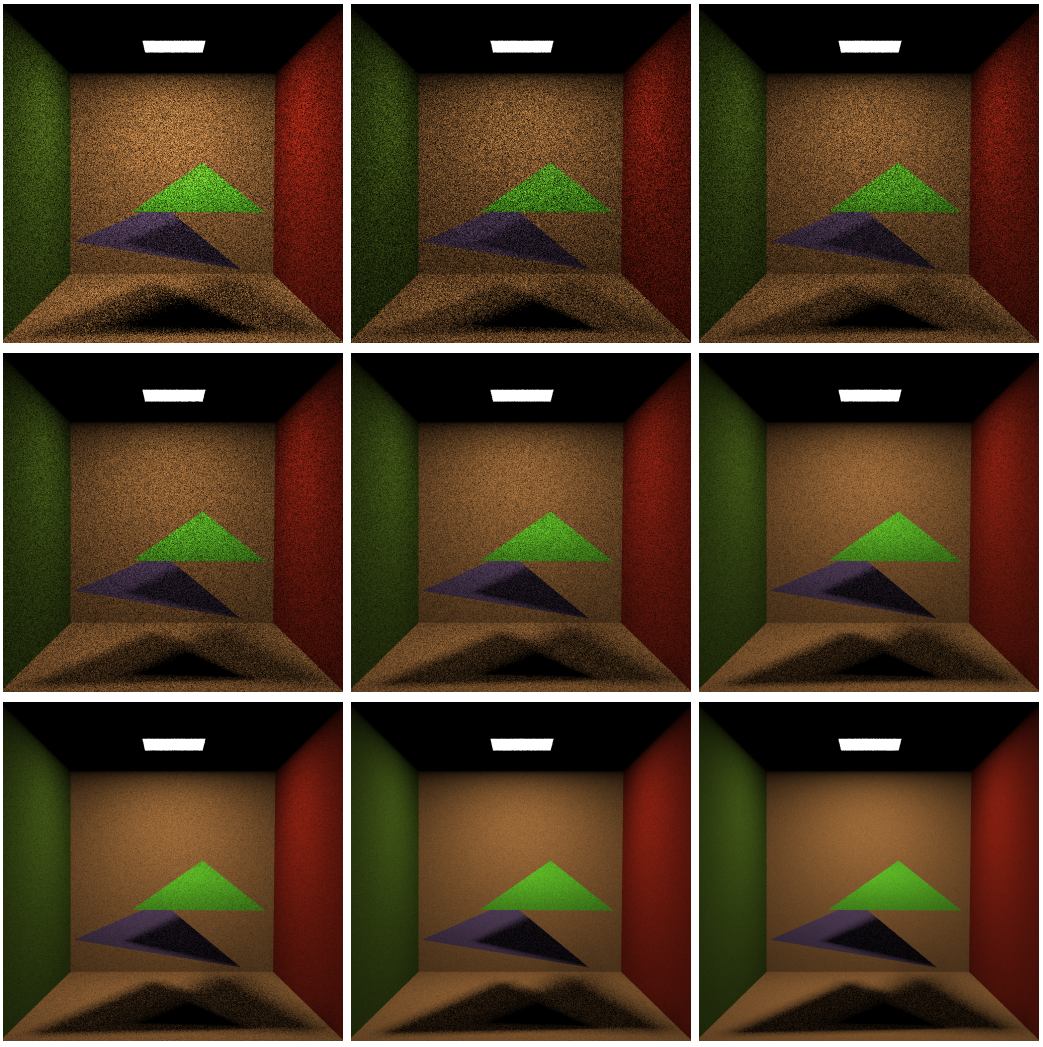
In deze integraal is de numerieke en niet de geometrische waarde van belang, vermits de numerieke waarde zal convergeren naar de juiste waarde. Figuur 2.2 is gerenderd met formule 2.6 en we zien dat de afbeelding convergeert naar het juiste resultaat wanneer het aantal schaduwstralen stijgt.

De bijdrage van stochastische visibiliteit is dat het aantal intersectietesten gemiddeld lager zal liggen. Voor de eerste twee termen hoeven we slechts één intersectie test uit te voeren en voor de derde term twee intersectietesten. Wanneer de kansen p_1, p_2 en p_3 gelijk zijn aan $1/3$ dan is de verwachte waarde van het aantal

intersectietesten gelijk aan:

$$E[\#_{intersecties}] = p_1 \cdot 1 + p_2 \cdot 1 + p_3 \cdot 2 \tag{2.15}$$
$$= \frac{1}{3} + \frac{1}{3} + \frac{2}{3} \tag{2.16}$$
$$= \frac{4}{3} \tag{2.17}$$

Voor de deterministische visibiliteitsevaluatie moeten we steeds 2 intersectietesten uitvoeren. Met stochastische visibiliteit zullen we dus gemiddeld 33% minder intersectietesten uitvoeren.



Figuur 2.2: Convergentie van de stochastische visibiliteit aan de hand van formule 2.6 voor 1, 2, 4, 8, 16, 32, 64, 128 en 256 schaduwstralen.

2.2 Uitbreiding naar meer primitieven

In deze sectie breiden we de theorie van de vorige sectie uit naar scenes met meer dan twee kandidaat blokkers. Hiervoor zullen we twee methodes bespreken. De eerste methode splitst de visibiliteitstermen V_A en V_B recursief op tot elke groep nog slechts één primitief bevat. De tweede methode groepeert de geometrie van de scene in twee disjuncte groepen A en B. Om de visibiliteiten V_A en V_B te bepalen, moeten we al de primitieven in respectievelijk groep A en groep B intersecteren.

2.2.1 Recursieve opdeling

Voor de volledige recursieve opdeling willen we dat elke groep nog slechts één kandidaat blokker bevat. Vermits het aantal kandidaat blokkers sterk kan variëren in de scene willen we een formule die eenvoudig is om te evalueren. Om zo een eenvoudig evaluatieschema te bekomen, gebruiken we de volgende gelijkheid die geldt voor booleaanse waarden:

$$a = 1 - \overline{a} \tag{2.18}$$

Indien we veronderstellen dat de verzameling $Z = \{z_1, z_2, z_i, ..., z_n\}$ de verzameling is van al de kandidaat blokkers tussen het punt x en y, dan kunnen we de visibiliteit met behulp van vergelijking 2.18 schrijven als:

$$V(x,y) = \prod_i^n V_{z_i}(x,y) \tag{2.19}$$

$$= \prod_i^n \left(1 - \overline{V_{z_i}}(x,y)\right) \tag{2.20}$$

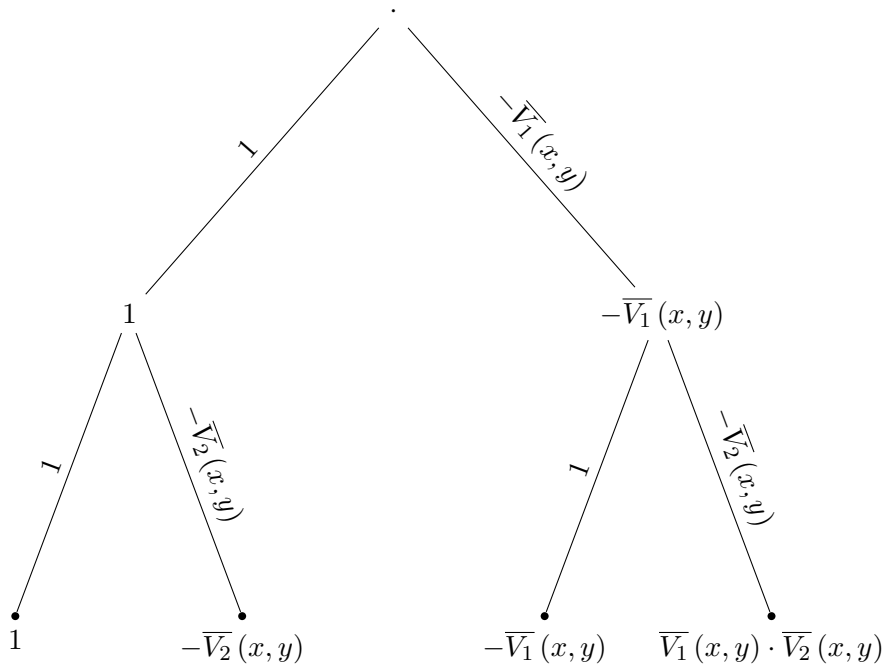
Door het product 2.19 om te vormen naar het product 2.20 met termen van de vorm $\left(1 - \overline{V_{z_i}}(x,y)\right)$ kunnen we de visibiliteit zeer eenvoudig evalueren door voor elk van de n termen in het product stochastisch ofwel de term 1 ofwel de term $-\overline{V_{z_i}}(x,y)$ te evalueren.

Dit evaluatieschema is afbeeld voor een scene met twee kandidaat blokkers in figuur 2.3. We beginnen in de wortel van de boom en we kiezen ofwel de term 1 ofwel de term $-\overline{V_1}(x,y)$. In de volgende stap maken we dan de keuze tussen de term 1 of de term $-\overline{V_2}(x,y)$. De eindknopen zijn het product van al onze keuzes in de boom. De kans om in eender welke eindknoop terecht te komen is gelijk aan $\frac{1}{2^n}$.

Het gemiddeld aantal intersecties in dit evaluatieschema is gelijk aan $n/2$. Voor elke van de n termen in vergelijking 2.20 hebben we immers de keuze tussen de 1 term die geen intersectie vergt en de term $-\overline{V_i}(x,y)$ die één intersectie vergt. Het benodigde aantal intersectietesten is dus gehalveerd.

Het gebruik van deze recursieve opdeling brengt wel een groot probleem met zich mee. De variantie van de Monte Carlo schatter is enorm. Herinner dat de variantie van een Monte Carlo schatter kan berekend worden met behulp van de volgende vergelijking:

$$\sigma^2 = \frac{1}{N} \int \left(\frac{f(x)}{p(x)} - I\right)^2 p(x)\, dx \tag{2.21}$$



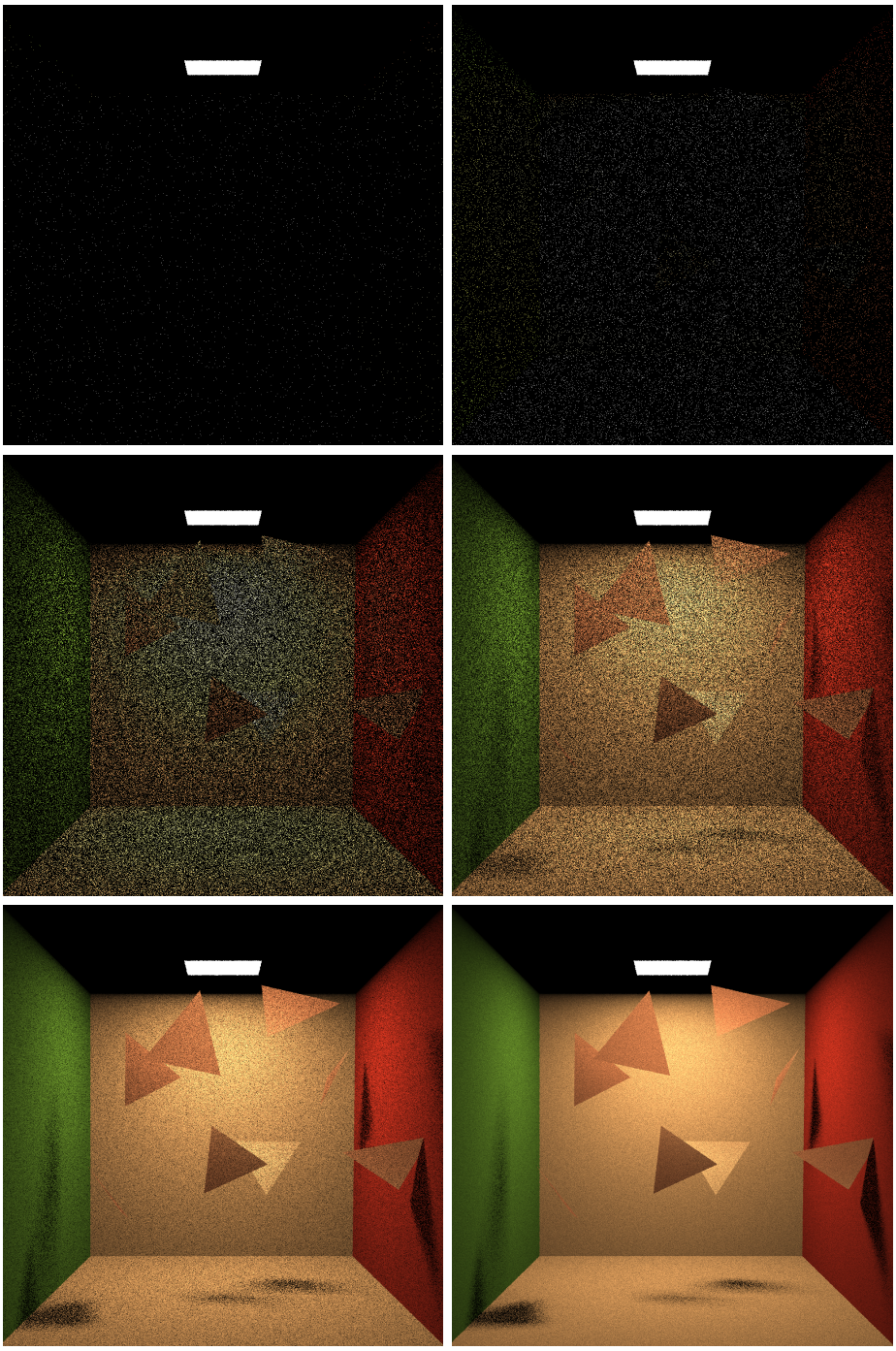
Figuur 2.3: Recursieve opsplitsing voor een scene met twee primitieven. We beginnen in de wortel van de boom en kiezen ofwel de term 1 of de term $-\overline{V_1}(x,y)$. In de volgende stap maken we dan weer een keuze tussen de term 1 en $-\overline{V_2}(x,y)$.

I is de exacte integraal en deze heeft een waarde binnen die ligt tussen 0 en 1. De kans $p(x)$ is gelijk aan $\frac{1}{2^n}$ waarbij n staat voor het aantal kandidaat blokkers voor de recursieve opsplitsing. Indien het aantal kandidaat blokkers n groot is, wat voor complexe scenes een redelijke aanname is, dan geldt dat $I \ll \frac{f(x)}{p(x)}$. De variantie zal dan bij benadering geschat kunnen worden als:

$$\sigma^2 \simeq \frac{1}{2^n N} \int \left(\frac{f(x)}{\frac{1}{2^n}} \right)^2 dx \tag{2.22}$$

$$\simeq \frac{2^n}{N} \int f(x)^2 \tag{2.23}$$

De variantie is dus evenredig met 2^n. Dit maakt de recursieve opdeling praktisch onbruikbaar voor complexe scenes waarbij het aantal mogelijke kandidaat blokkers groot is. Figuur 2.4 toont een scene met 8 kandidaat blokkers. Uit deze figuur is duidelijk dat de figuur convergeert naar het juiste resultaat, maar de variantie in de afbeeldingen is enorm en er zijn een groot aantal schaduwstralen nodig om deze variantie te drukken.



FIGUUR 2.4: Scene met 8 kandidaat blokkers. De afbeeldingen zijn gerenderd met behulp van de recursieve formule met 1, 8, 64, 256, 1024 en 4096 schaduwstralen.

2.2.2 Meerdere primitieven per groep

Zoals bleek in sectie 2.2.1 stijgt de variantie exponentieel met het aantal kandidaat blokkers wanneer we de visibiliteit recursief opdelen in scenes met meerdere primitieven. In deze sectie behouden we de oorspronkelijke formule voor de stochastische visibiliteit uit sectie 2.1. In plaats van de formule recursief toe te passen zullen we de primitieven in de scene onderverdelen in twee disjuncte groepen A en B. Indien de verzameling $Z = \{z_1, z_2, z_i, ..., z_n\}$ de verzameling van kandidaat blokkers is tussen de punten x en y dan stellen we V_A en V_B gelijk aan:

$$V_A(x,y) = \sum_{i=1}^{n/2-1} V_{z_i}(x,y) \tag{2.24}$$

$$V_B(x,y) = \sum_{i=n/2}^{n} V_{z_i}(x,y) \tag{2.25}$$

Iedere groep bevat dus $n/2$ primitieven. Door het gebruik van stochastische visibiliteit zal het benodigde aantal intersecties dalen, zoals besproken in sectie 2.1.

$$E[\#_{intersecties}] = p_1 \cdot \frac{n}{2} + p_2 \cdot \frac{n}{2} + p_3 \cdot n \tag{2.26}$$

$$= \frac{n}{6} + \frac{n}{6} + \frac{n}{3} \tag{2.27}$$

$$= \frac{2}{3} \cdot n \tag{2.28}$$

Deterministische visibiliteit vereist steeds n intersecties. Gemiddeld zullen we dus 33% minder intersecties doen met behulp van stochastische visibiliteit. We doen dus slechter dan de recursieve formule waar we gemiddeld 50% van de intersecties besparen.

De variantie van dit evaluatieschema zal ook veel lager zijn dan de variantie in het recursieve schema. Vermits $p(x) = 1/3$ is de variantie gelijk aan:

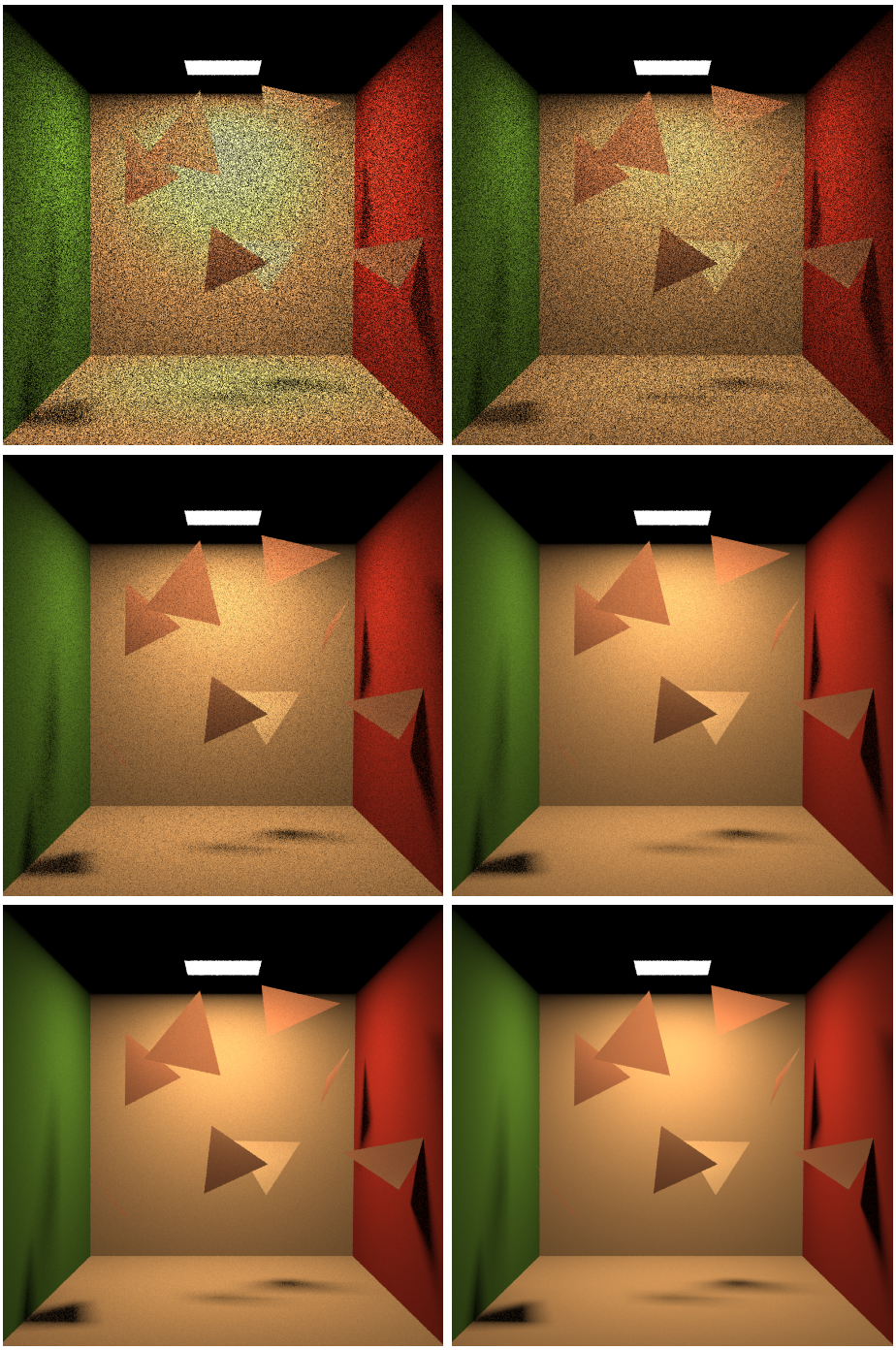
$$\sigma^2 = \frac{1}{N} \int \left(\frac{f(x)}{p(x)} - I \right)^2 p(x)\, dx \tag{2.29}$$

$$= \frac{1}{3N} \int (3f(x) - I)^2\, dx \tag{2.30}$$

Wanneer we vergelijking 2.30 en vergelijking 2.23 naast elkaar zetten, dan zien we dat de variantie van vergelijking 2.30 kleiner is dan de variantie van vergelijking 2.23 voor grote n:

$$\frac{1}{3N} \int (3f(x) - I)^2\, dx \ll \frac{2^n}{N} \int f(x)^2 \tag{2.31}$$

Figuur 2.5 toont de scene waarbij de visibiliteit geëvalueerd wordt door de 8 kandidaat blokkers willekeurig in twee groepen van 4 te verdelen. Zoals verwacht zien we dat de variantie sterk gedaald is ten opzichte van de figuren met het recursieve schema 2.4.



FIGUUR 2.5: Scene met 8 kandidaat blokkers. De acht blokkers zijn willekeurig in twee groepen van 4 blokkers onderverdeeld. De afbeeldingen zijn gerenderd met 1, 8, 64, 256, 1024 en 4096 schaduwstralen.

2.3 Alternatieve decomposities

Vergelijking 2.6 is slechts één van de mogelijke manieren waarmee een product omgevormd kan worden tot een som. In deze sectie leiden we enkele andere formules af waarvan we zullen aantonen dat ze sneller convergeren.

2.3.1 Merkwaardig product #1

In sectie 2.1 maakten we gebruik van het merkwaardig product $(1-a)(1-b) = 1-a-b+ab$ om de visibiliteit op te splitsen in verschillende termen. We kunnen deze formule echter nog verder veralgemenen door de -1 term op te splitsen en te herverdelen over de overige drie termen. Op deze manier komen we tot de volgende formule voor de visibiliteit:

$$V(x,y) = V_A(x,y)\,V_B(x,y) \tag{2.32}$$
$$= V_A(x,y) - \alpha + V_B(x,y) - \beta + \overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - \gamma \tag{2.33}$$

Om de visibiliteit te schatten, kiezen we dus één van de volgende drie termen:

$$\tilde{V}(x,y) = \begin{cases} \frac{V_A(x,y)-\alpha}{p_1} & \text{met kans } p_1 \\ \frac{V_B(x,y)-\beta}{p_2} & \text{met kans } p_2 \\ \frac{V_A(x,y)\cdot V_B(x,y)-\gamma}{p_3} & \text{met kans } p_3 \end{cases} \tag{2.34}$$

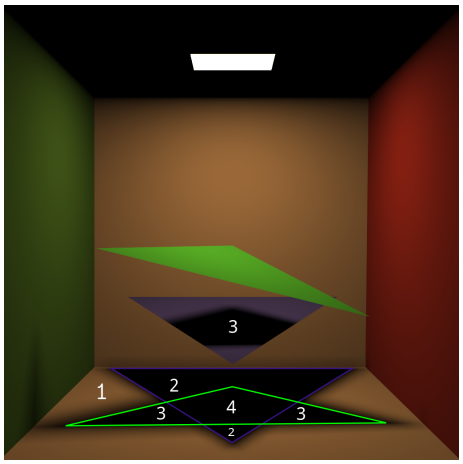
Door verschillende waardes voor de termen α, β en γ te gebruiken, kunnen we de variantie in verschillende gebieden van de scene verminderen. We onderscheiden vier gebieden in de scene, aangeduid in figuur 2.6. We zullen voor ieder van deze vier gebieden de waardes van α, β en γ berekenen die de variantie in dat gebied zal reduceren tot 0.

Gebied 1

Dit is het gebied in de scene dat compleet belicht wordt. In dit gebied weten we dat visibiliteit $V(x,y)$ gelijk is aan 1 en dat $V_A(x,y)$ en $V_B(x,y)$ dus ook gelijk moeten zijn aan 1. Om de variantie gelijk te stellen aan nul in dit gebied moeten de drie termen uit 2.34 dus evalueren tot 1. Daarvoor moeten we het volgende stelsel oplossen:

$$(V_A(x,y) - \alpha)/p_1 = 3(1-\alpha) = 1 \tag{2.35}$$
$$(V_B(x,y) - \beta)/p_2 = 3(1-\beta) = 1 \tag{2.36}$$
$$\left(\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - \gamma\right)/p_3 = 3(-\gamma) = 1 \tag{2.37}$$

De oplossing van dit stelsel is $\alpha = 2/3$, $\beta = 2/3$ en $\gamma = -1/3$.



FIGUUR 2.6: De verschillende gebieden van de scene. Het nummer 1 geeft gebieden aan die compleet belicht zijn. Het nummer 2 en het nummer 3 geven respectievelijk de schaduwgebieden aan van de paarse en de groene driehoek. Het nummer 4 toont het gebied van de scene dat zowel door de groene als door de paarse driehoek in schaduw ligt.

Gebied 2

Dit is het gebied dat enkel in schaduw ligt door de paarse driehoek. De visibiliteit $V(x,y)$ gaat hier nul zijn, omdat $V_A(x,y)$ gelijk zal zijn aan 0 en $V_B(x,y)$ gelijk zal zijn aan 1. Met behulp van het volgend stelsel kunnen we de variantie gelijkstellen aan nul:

$$\begin{aligned}(V_A(x,y)-\alpha)/p_1 &= 3(-\alpha) = 0\\ (V_B(x,y)-\beta)/p_2 &= 3(1-\beta) = 0\\ \left(\overline{V_A(x,y)}\cdot\overline{V_B(x,y)}-\gamma\right)/p_3 &= 3(-\gamma) = 0\end{aligned}$$

De oplossing van dit stelsel is $\alpha = 0$, $\beta = 1$ en $\gamma = 0$.

Gebied 3

Dit gebied is het gebied dat enkel in schaduw ligt door de groene driehoek. De visibiliteit $V(x,y)$ gaat hier nul zijn, omdat $V_B(x,y)$ gelijk zal zijn aan 0 en $V_A(x,y)$ gelijk zal zijn aan 1. Met behulp van het volgend stelsel kunnen we de variantie gelijkstellen aan nul:

$$\begin{aligned}(V_A(x,y)-\alpha)/p_1 &= 3(1-\alpha) = 0\\ (V_B(x,y)-\beta)/p_2 &= 3(-\beta) = 0\\ \left(\overline{V_A(x,y)}\cdot\overline{V_B(x,y)}-\gamma\right)/p_3 &= 3(-\gamma) = 0\end{aligned}$$

De oplossing voor dit stelsel is $\alpha = 1$, $\beta = 0$ en $\gamma = 0$.

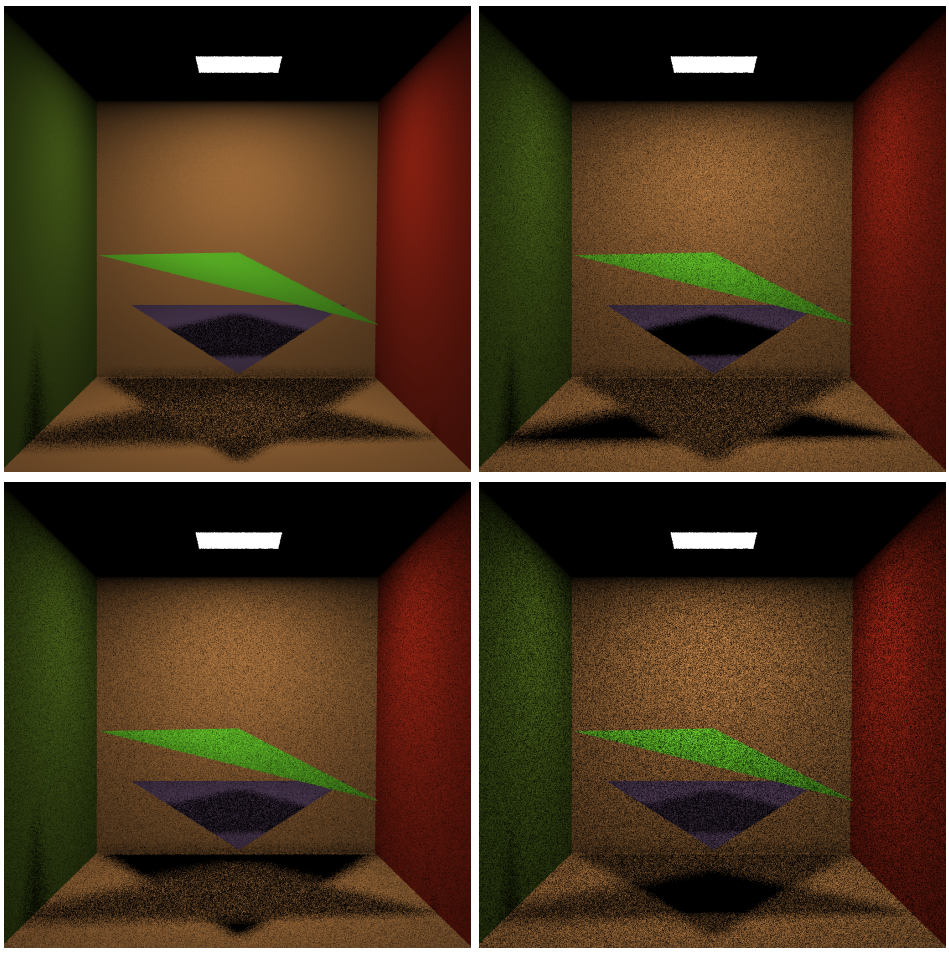
Gebied 4

Dit gebied ligt zowel door de groene als door de paarse driehoek in schaduw. Zowel $V_A(x, y)$ als $V_B(x, y)$ zullen hier nul zijn. Met behulp van het volgende stelsel kunnen we de variantie in dit gebied gelijkstellen aan nul:

$$\begin{aligned} (V_A(x,y) - \alpha)/p_1 &= 3(-\alpha) = 0 \\ (V_B(x,y) - \beta)/p_2 &= 3(-\beta) = 0 \\ \left(\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - \gamma\right)/p_3 &= 3(1-\gamma) = 0 \end{aligned}$$

De oplossing van dit stelsel leidt ons tot de oorspronkelijke formule 2.6 met $\alpha = 0$, $\beta = 0$ en $\gamma = 1$.

Figuur 2.7 toont de scene met de twee driehoeken gerenderd met de optimale parameters voor de vier gebieden.



Figuur 2.7: Stochastische visibiliteit gerenderd met de optimale parameters voor respectievelijk gebied 1, 2, 3 en 4.

2.3.2 Merkwaardig product #2

We kunnen ook een ander merkwaardig product gebruiken om het product van visibiliteiten om te zetten naar een som. Met behulp van het merkwaardig product $(a-b)^2 = a^2 - 2ab + b^2$ kunnen we de visibiliteit schrijven als:

$$V(x,y) = V_A(x,y) \cdot V_B(x,y) \tag{2.38}$$

$$= \frac{V_A(x,y)^2}{2} + \frac{V_B(x,y)^2}{2} - \frac{(V_A(x,y) - V_B(x,y))^2}{2} \tag{2.39}$$

$$= \frac{V_A(x,y)}{2} + \frac{V_B(x,y)}{2} - \frac{(V_A(x,y) - V_B(x,y))^2}{2} \tag{2.40}$$

We kunnen verwachten dat de variantie van deze ontbinding lager is dan de variantie van formule 2.6 vanwege de factor $\frac{1}{2}$. Een eigenschap van variantie is immers dat:

$$var(aX) = a^2 var(X) \tag{2.41}$$

Indien we kijken naar de waardes van de stochastische evaluaties in tabel 2.2, dan zien we dat de waardes van al de termen in de tabel in absolute waarde kleiner zijn dan de waardes van tabel 2.1. De waardes in tabel 2.2 liggen daardoor ook dichter bij de exacte waardes. Hierdoor zal de term $\left(\frac{f(x)}{p(x)} - I\right)^2$ in de formule voor de variantie 2.21 ook kleiner zijn, wat de variantie verder naar beneden drukt.

Exacte waarden			**Stochastische evaluatie**		
V_A	V_B	$V_A \cdot V_B$	$\frac{3}{2}V_A$	$\frac{3}{2}V_B$	$-\frac{3}{2}(V_A - V_B)^2$
0	0	0	0	0	0
0	1	0	0	3/2	-3/2
1	0	0	3/2	0	-3/2
1	1	1	3/2	3/2	0

TABEL 2.2: Stochastische evaluatie van de visibiliteit met behulp van formule 2.40. De drie linkse kolommen geven de exacte waardes voor de vier mogelijke visibiliteitsevents. De drie rechtse kolommen geven de waarde weer in het geval we slechts één term evalueren.

2.3.3 Binomiaal formule

In deze subsectie gebruiken we een complexer merkwaardig product om een product te ontbinden in een som. We gebruiken hiervoor de binomiaal formule:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i} \tag{2.42}$$

$$= a^n + b^n + \sum_{i=1}^{n-1} \binom{n}{i} a^i b^{n-i} \tag{2.43}$$

$$= a + b + (2^n - 2)\, ab \tag{2.44}$$

$$ab = -\frac{a}{2^n-2} - \frac{b}{2^n-2} + \frac{(a+b)^n}{2^n-2} \tag{2.45}$$

Uit formule 2.42 kunnen we de eerste term en de laatste term van de som afzonderen, wat resulteert in vergelijking 2.43. Voor booleaanse waardes geldt dat $a^n = a$. Verder gebruiken we $\sum_{i=0}^{n} \binom{n}{i} = 2^n$ om 2.43 te herschrijven tot 2.44.

Wanneer we de binomiaal formule toepassen op de visibiliteit bekomen we de volgende formule:

$$V(x,y) = V_A(x,y)\, V_B(x,y) \tag{2.46}$$

$$= -\frac{V_A(x,y)}{2^n-2} - \frac{V_B(x,y)}{2^n-2} + \frac{(V_A(x,y) + V_B(x,y))^n}{2^n-2} \tag{2.47}$$

Hierbij is de term n een term die we vrij mogen kiezen zolang $n > 1$. Bij de keuze van n moeten we met twee conflicterende voorwaarden rekening houden. Ten eerste willen we n zo groot mogelijk kiezen. Hoe groter de term n, des te kleiner de termen worden in absolute waarden. Hierdoor liggen de termen dichter bij de exacte visibiliteitswaarden, waardoor we de variantie beperken. Aan de andere kant willen we n zo klein mogelijk houden, omdat anders de eerste twee termen zo klein worden dat ze geen bijdrage meer hebben tot de integraal. Experimenteel bleek $n = 8$ een goede balans te geven.

Exacte waarden			**Stochastische evaluatie**		
V_A	V_B	$V_A \cdot V_B$	$\frac{3}{254} V_A$	$\frac{3}{254} V_B$	$\frac{3}{254} (V_A + V_B)^8$
0	0	0	0	0	0
0	1	0	-3/254	0	3/254
1	0	0	0	-3/254	3/254
1	1	1	-3/254	-3/254	768/254

Tabel 2.3: Stochastische evaluatie van de visibiliteit met behulp van formule 2.47. De drie linkse kolommen geven de exacte waardes voor de vier mogelijke visibiliteitsevents. De drie rechtse kolommen geven de waarde weer in het geval we slechts één term evalueren.

2.3.4 Vergelijking van de ontbindingen

Figuur 2.8 toont de scene met de twee driehoeken gerenderd met de drie stochastische visibiliteit formules 2.34, 2.40 en 2.47. Figuur 2.9 toont een close-up van figuur 2.8.

De ontbinding met merkwaardig product #1 laat ons toe om de variantie in een van de regio's van de scene te reduceren tot nul. In deze figuur hebben we ervoor gekozen om de variantie in gebied 4 (het gebied dat door beide driehoeken in schaduw ligt) naar nul te brengen. We zien echter dat in dit geval de ontbinding met merkwaardig product #2 het nog steeds beter doet. Voor de ontbinding met het tweede merkwaardig product is de variantie immers ook nul in het gebied dat door beide driehoeken in schaduw ligt. Daarnaast zien we in de close-ups van figuur 2.9 de variantie ook veel kleiner is in de volledig belichte gebieden wanneer we de ontbinding #2 kiezen.

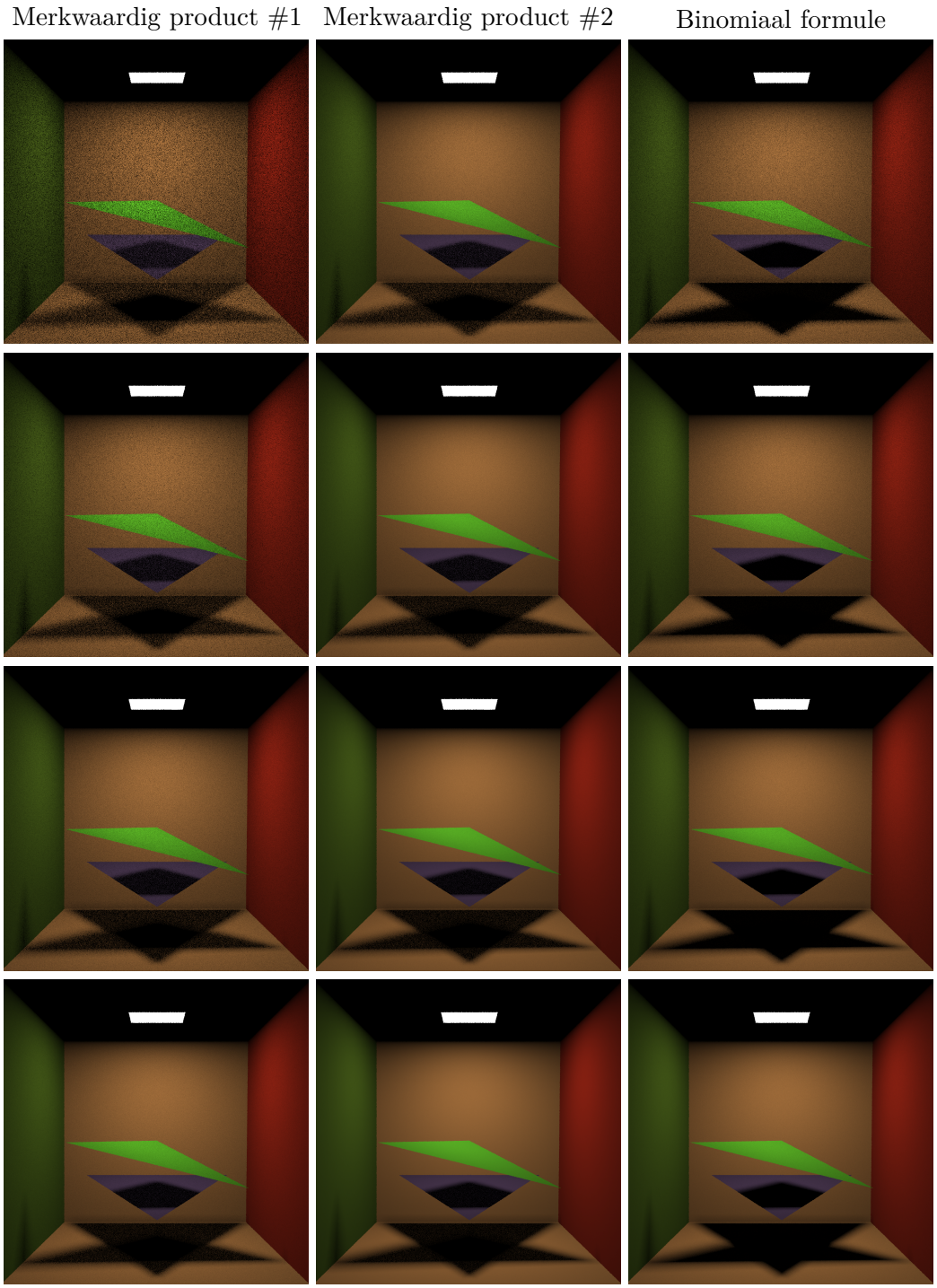
De binomiaal formule geeft echter de beste visuele resultaten. In tegenstelling tot de ontbinding met merkwaardig product #1 en #2 convergeert de volledige schaduwregio zelfs al bij een laag aantal schaduwstralen. De convergentie van de volledig belichte gebieden in de scenes is echter beter wanneer we ontbinding #2 gebruiken.

2.4 Besluit

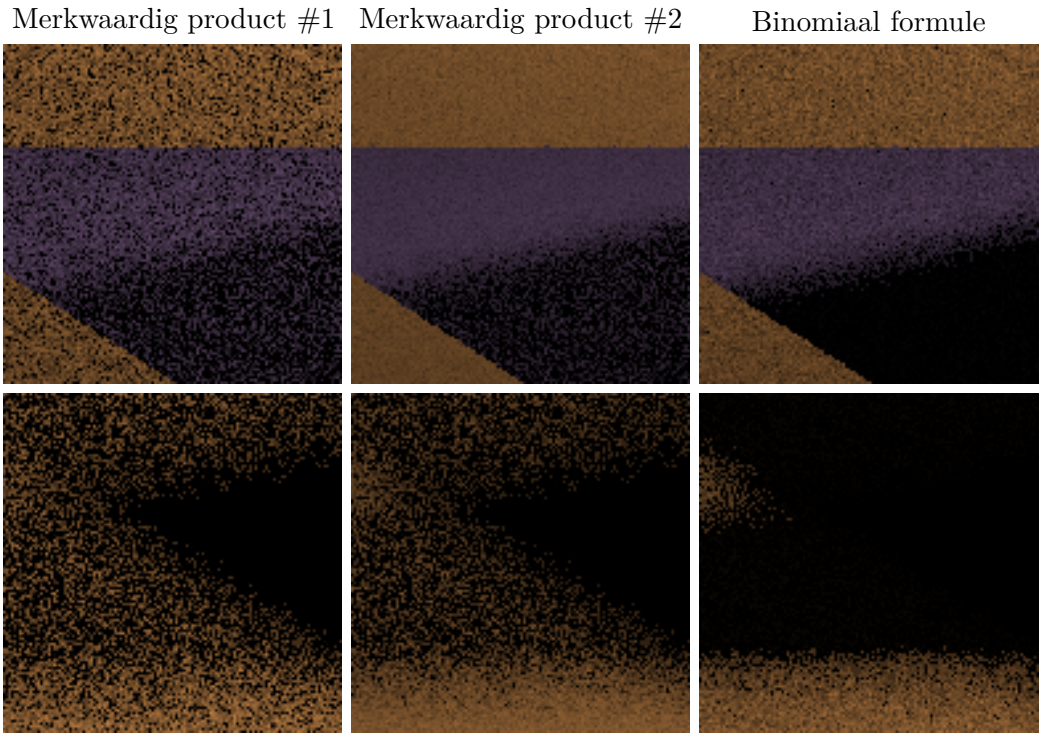
In dit hoofdstuk hebben we de theorie van stochastische visibiliteit uitgelegd. Door het product van de visibiliteiten te herschrijven als een som, konden we de som benaderen door slechts één van de termen te selecteren en te evalueren. We toonden dat de variantie stijgt door de stochastische visibiliteitsevaluatie, maar het benodigde aantal intersectietesten daalde.

Verder kunnen we in dit hoofdstuk besluiten dat het recursief toepassen van de visibiliteitsevaluatie niet praktisch is vanwege de enorme toename in de variantie. In een praktisch belichtingsalgoritme zullen we de kandidaat blokkers dus steeds in twee groepen opdelen.

Finaal hebben we aangetoond dat de keuze van de decompositie voor de stochastische visibiliteit een grote invloed heeft op de variantie. De visibiliteitsdecompositie op basis van de binomiaal formule introduceert hierbij de minste variantie.



Figuur 2.8: Vergelijking van de verschillende decomposities. De eerste kolom is gerenderd met vergelijking 2.6, de tweede kolom is gerenderd met 2.40 en de derde kolom is gerenderd met 2.47. Uit deze figuur zien we dat de binomiaal formule over he algemeen het beste convergentie gedrag heeft. De afbeeldingen in iedere rij zijn gerenderd met respectievelijk 16, 64, 256 en 1024 schaduwstralen.



FIGUUR 2.9: Close-up van figuur 2.8 met de verschillende formules. De linkse figuur is gerenderd met merkwaardig product #1. De middelste figuur is gerenderd met merkwaardig product #2. De rechtse figuur is gerenderd met de binomiaal formule.

Hoofdstuk 3

De occlusion map

Om de theorie van hoofdstuk 2 toe te kunnen passen in een praktisch algoritme hebben we een datastructuur nodig die al de kandidaat blokkers tussen een punt en een lichtbron in de scene kan vinden. In dit hoofdstuk introduceren we de *occlusion map*, een datastructuur die toelaat een benadering te krijgen van de verzameling kandidaat blokkers tussen een punt en de lichtbron. Deze data structuur is een uitbreiding op de photon map [5], een datastructuur die reeds vaak gebruikt wordt om de globale belichting te berekenen. In de eerste sectie zullen we uitleggen hoe we de photon map construeren en hoe deze gebruikt kan worden om de globale belichting te berekenen. Vervolgens breiden we de photon map uit zodat deze de kandidaat blokkers tussen een lichtbron en een punt kan berekenen.

3.1 Photon mapping

Photon mapping is een algoritme dat bestaat uit twee aparte fases om de globale belichting te berekenen. In een eerste fase worden licht deeltjes of photonen vanaf de lichtbronnen doorheen de ruimte verspreid. In de tweede fase wordt de belichting van een punt in de ruimte benaderd door de belichtingsinformatie te gebruiken van de dichtstbijzijnde photonen rond het punt. In de volgende subsecties zullen we deze fases meer in detail bespreken.

3.1.1 Constructie van de photon map

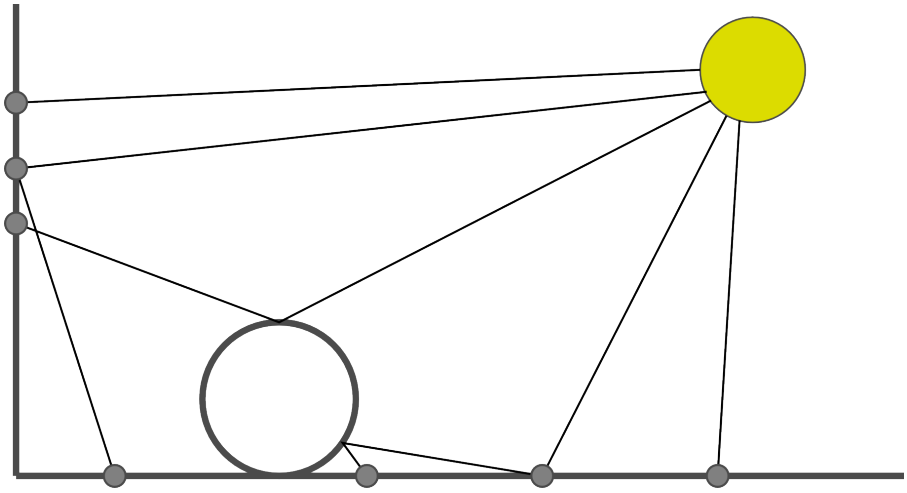
Voor de constructie van de photon map moeten er photonen doorheen de ruimte verspreid worden. Het verdelen van de photonen doorheen de ruimte is gelijkaardig aan het path tracing algoritme uit sectie 1.5. We hergebruiken de path tracing mechanismen om de photonen te distribueren waarbij de paden nu beginnen vanaf de lichtbron. Na de distributie worden de photonen verzameld in een ruimtelijke acceleratiestructuur, zodat we tijdens het renderen efficiënt de dichtstbijzijnde photonen rond een punt kunnen zoeken.

Photon distributie

De photonen worden vanaf de lichtbronnen verdeeld in de ruimte. Om de photonen in de ruimte te distribueren, volgen we stralen $r\,(x_l \rightarrow \Psi)$ die vertrekken vanuit een willekeurige positie en richting vanaf de lichtbron. We volgen deze stralen doorheen de ruimte tot in hun eerste intersectie punt x. De materiaaleigenschappen in het punt x bepalen of er een photon gecreëerd wordt en of er verdere verstrooiing van het photon plaatsvindt.

- **diffuus materiaal**: op een diffuus materiaal wordt er steeds een photon aangemaakt. Dit photon slaat zijn positie x, zijn richting Ψ en zijn flux Ω op, waarbij de flux de hoeveelheid licht is dat invalt op de positie x. Verder zullen we met behulp van Russische roulette kiezen of er verstrooiing of absorptie gaat optreden. Wanneer er verstrooiing plaatsvindt, zullen we in een willekeurige richting vanuit het punt x een photon proberen te distribueren. Bij absorptie stoppen we de recursieve verdeling van de photonen op het huidige pad.
- **speculaire materiaal**: de straal die we volgden vanaf de lichtbron wordt weerkaatst in de perfect spiegelende richting en recursief verder doorheen de scene gevolgd.
- **refractief materiaal**: hierbij zullen we stochastisch kiezen tussen de perfect weerkaatste richting en de gebroken richting om de straal vanaf de lichtbron recursief doorheen de scene te volgen.

Figuur 3.1 illustreert hoe de photonen doorheen een diffuse ruimte met een speculaire bal verspreid worden.



Figuur 3.1: Distributie van de photonen. Deze figuur illustreert hoe photonen doorheen de ruimte verspreid worden vanaf de lichtbron. De bol in de scene is speculair waardoor er recursief stralen in de spiegelende richting gevolgd moeten worden.

Acceleratiestructuur

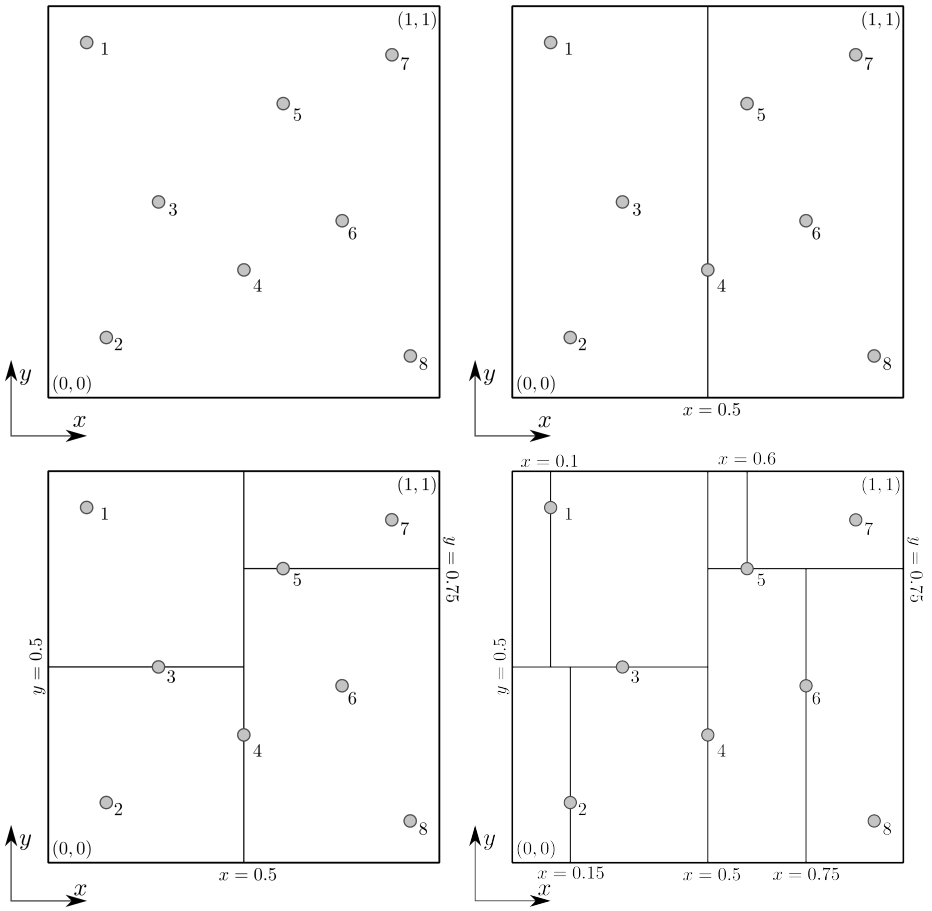
Wanneer al de photonen doorheen de ruimte verspreid zijn, worden ze opgeslagen in een ruimtelijke acceleratiestructuur die toelaat om zeer snel al de photonen in de buurt van een bepaald punt in de ruimte te vinden. De datastructuur die hier vaak voor gebruikt wordt is een gebalanceerde kd-boom.

Een kd-boom is een binaire boomstructuur waarin elke knoop de ruimte in twee delen met een gelijk aantal photonen verdeelt. De constructie van de kd-boom begint door de photonen te sorteren volgens de x-, y- of z-dimensie. We verdelen de ruimte in twee met behulp van een vlak, in de gekozen dimensie, op de positie van het middelste photon in de gesorteerde lijst. Hierdoor wordt de ruimte verdeeld in twee helften die elk de helft van de photonen bevat. Bovenstaande procedure wordt dan recursief toegepast op beide helften totdat er nog maar één photon in elke helft overblijft. De vlakken plaatsen we achtereenvolgens in de x-, y- en z-dimensie.

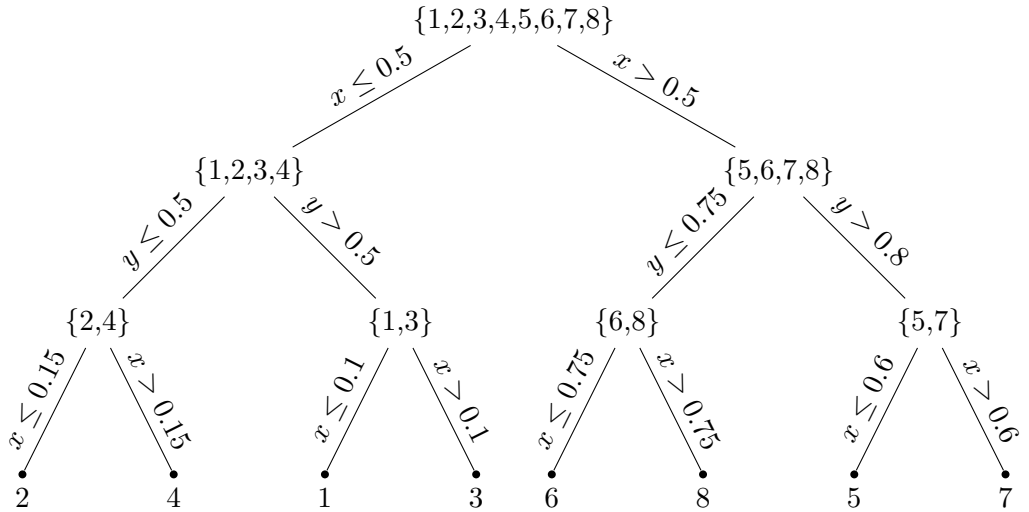
Het opbouwen van een kd-boom is afgebeeld in figuur 3.2. Hierbij bouwen we de kd-boom voor 8 photonen en beginnen we door de photonen te sorteren volgens de x-richting. Hierbij is photon 4 op $x = 0.5$ het middelste photon en we verdelen de ruimte in twee helften op deze positie. De linkerhelft bevat de photonen waarvan de x-coördinaat kleiner is dan of gelijk aan 0.5, de rechterhelft bevat de photonen waarvan de x-coördinaat groter is dan 0.5. We zetten deze procedure recursief verder voor beide helften waarbij we afwisselend volgens de x- en y-richting sorteren. De resulterende kd-boom is afgebeeld in figuur 3.3.

Om de dichtstbijzijnde photonen rond een punt te bepalen gebruiken we de kd-boom om grote delen van de ruimte over te slaan. Indien we al de photonen binnen de bol met straal r en oorsprong p willen vinden, dalen we recursief af in de kd-boom. Elke knoop definieert een vlak dat de ruimte in twee helften deelt. Indien de bol zich slechts in één van de twee helften bevindt, hoeven we enkel verder af te dalen in de knoop van dat vlak.

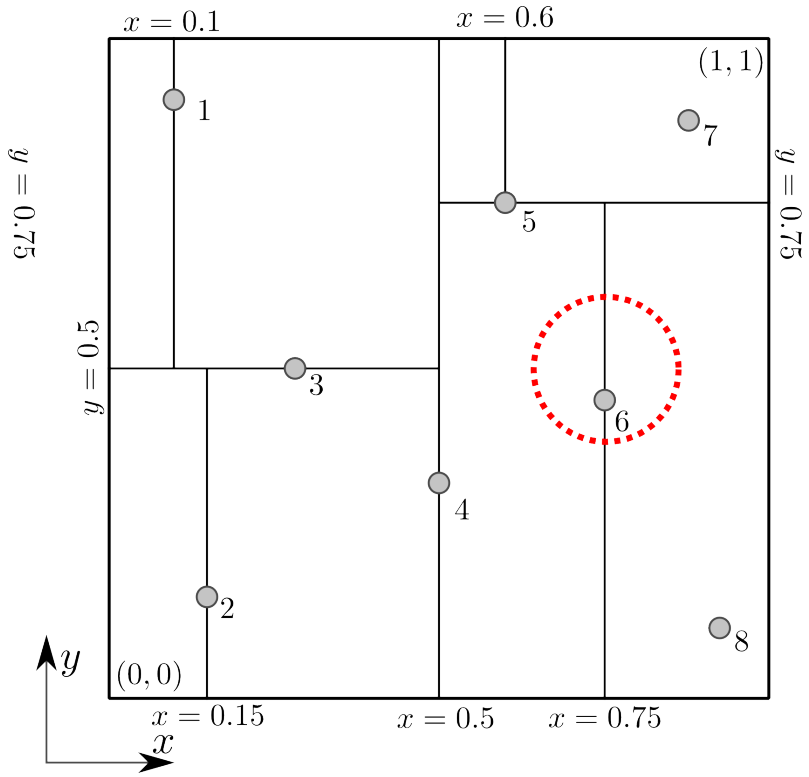
Het zoeken van photonen rond het punt $(0.75, 0.5)$ met een straal van 0.1 is geïllustreerd in figuren 3.4 en 3.5. In de eerste knoop berekenen we of de cirkel links of rechts van het vlak $x = 0.5$ ligt. Omdat de bol rechts van dit vlak ligt, weten we dat de photonen 1, 2, 3 en 4 nooit binnen de cirkel kunnen vallen. Daarom hoeven we de linker knoop van de kd-boom niet af te dalen. In de volgende knoop moeten we berekenen of de bol boven of onder het vlak $y = 0.75$ ligt. Omdat de bol onder het vlak ligt weten we dat photonen 5 en 7 nooit binnen de cirkel kunnen liggen. Finaal moeten we berekenen of de cirkel links of rechts van het vlak $x = 0.75$ ligt. Omdat de cirkel zich zowel in het linker als het rechter vlak bevindt, moeten we photon 6 en 8 testen of ze binnen de cirkel liggen. Op deze manier vinden we dat photon 6 het dichtstbijzijnde photon is.



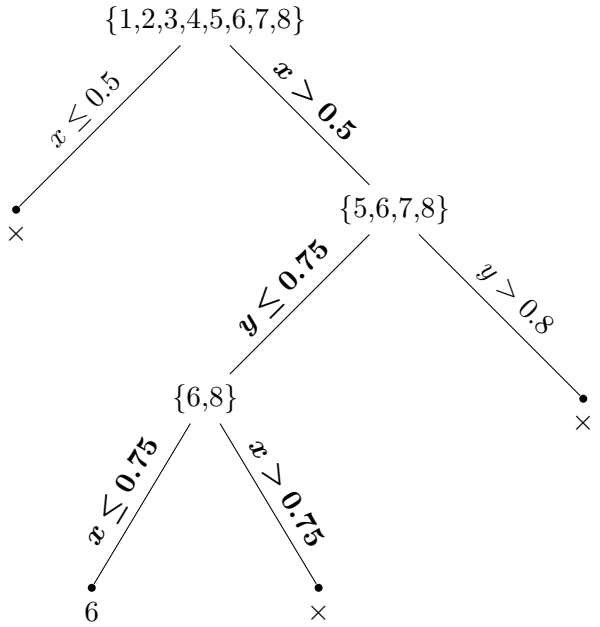
Figuur 3.2: Constructie van de kd-boom. In dit voorbeeld wordt er een kd-boom opgebouwd voor 8 photonen waarbij we we eerst volgens de x richting sorteren.



Figuur 3.3: Voorstelling van de kd-boom voor de photonen uit figuur 3.2.



FIGUUR 3.4: Illustratie van een kd-tree look-up met straal 0.1 rond het punt $(0.75, 0.5)$.



FIGUUR 3.5: Het doorlopen van de kd-boom voor de photon look-up uit figuur 3.4.

3.1.2 Renderen met de photon map

De rendering vergelijking kan opgelost worden met behulp van de photon map.

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_\Omega f_r(x, \Theta \leftrightarrow \Psi)\, L(x \leftarrow \Psi) \cos(N, \Psi)\, d\omega_\Psi \tag{3.1}$$

De invallende radiantie $L(x \leftarrow \Psi)$ kan benaderd worden met behulp van de photon map door de volgende relatie:

$$L(x \leftarrow \Psi) = \frac{d^2\Omega(x, \Psi)}{\cos(N, \Psi)\, d\Psi dA} \tag{3.2}$$

Hierbij staat de grootheid Ω voor de hoeveelheid flux die vanuit de infinitesimale richting $d\Psi$ invalt op een infinitesimaal oppervlak dA met positie x. Indien we de relatie invullen in de rendering vergelijking vereenvoudigt deze zich tot:

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_\Omega f_r(x, \Theta \leftrightarrow \Psi) \frac{d^2\Omega(x, \Psi)}{\cos(N, \Psi)\, d\Psi dA} \cos(N, \Psi)\, d\omega_\Psi \tag{3.3}$$

$$= L_e(x \to \Theta) + \int_\Omega f_r(x, \Theta \leftrightarrow \Psi) \frac{d^2\Omega(x, \Psi)}{dA} \tag{3.4}$$

De hoeveelheid flux kunnen we benaderen met de flux van de n dichtstbijzijnde photonen rond het punt x.

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_\Omega f_r(x, \Theta \leftrightarrow \Psi) \frac{d^2\Omega(x, \Psi)}{dA} \tag{3.5}$$

$$= L_e(x \to \Theta) + \sum_{i=1}^{n} f_r(x, \Theta \leftrightarrow \Psi_i) \frac{\Omega_i(x, \Psi_i)}{\Delta A} \tag{3.6}$$

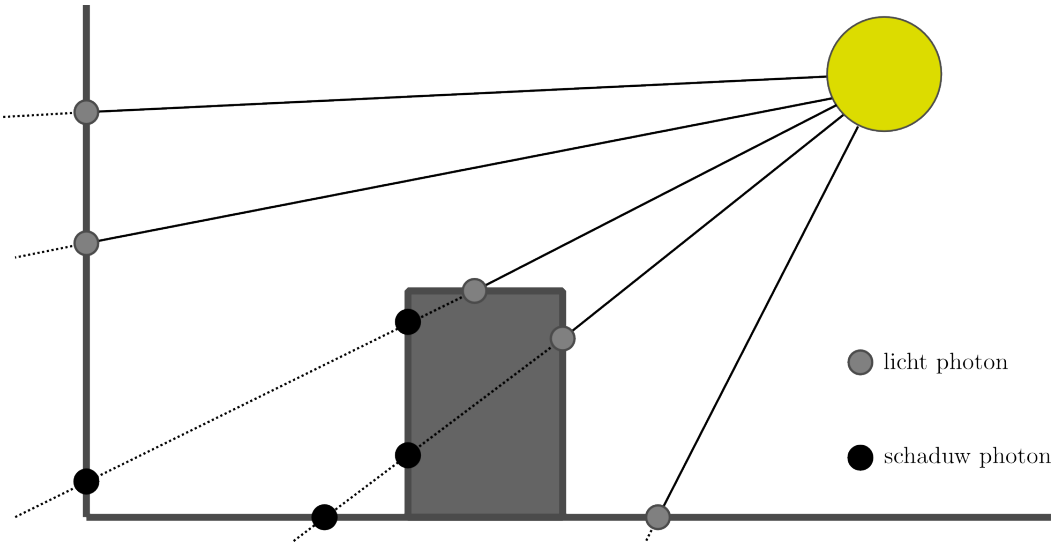
Hierbij staat Ω_i voor de flux opgeslagen in photon i en is Ψ_i de richting vanwaar het photon komt. De oppervlakte ΔA wordt benaderd door πr^2, waarbij r de straal is waarin we photonen zoeken rond het punt x. Alhoewel we de photonen in een bol rond het punt x zoeken, gebruiken we de oppervlakte van een cirkel als benadering voor ΔA omdat we aannemen dat al de photonen op een infinitesimaal oppervlak liggen.

De photon map zal echter zelden gebruikt worden om de belichting direct te visualiseren. De reden hiervoor is dat het verzamelen van de photonen de fijne details zoals schaduw randen en texturen doen vervagen. Daarom zal men de photon map meestal gebruiken in combinatie met ray tracing. Daarbij wordt de directe belichting gerenderd met behulp van path tracing, terwijl de indirecte belichting berekend wordt met behulp van de photon map.

3.2 Schaduw photonen

Zoals eerder besproken wordt de directe belichting zelden met de photon map gerenderd. Door een simpele uitbreiding op de photon map kunnen we echter het aantal schaduwstralen dat voor de directe belichting door de scene gevolgd moet worden reduceren tot 80% [5].

Naast photonen die licht opslaan, breiden we de photon map uit met schaduw photonen die de schaduw posities in de scene aanduiden. Wanneer we een licht photon creëren, volgen we de straal verder doorheen het oppervlak. Op elke volgende intersectie van deze straal creëren we een schaduw photon. Dit proces is geïllustreerd in figuur 3.6.



FIGUUR 3.6: De photon map uitgebreid met schaduw photonen. Wanneer we een licht photon aanmaken, volgen we de straal verder doorheen het oppervlak en creëren we een schaduw photon op elke volgende intersectie

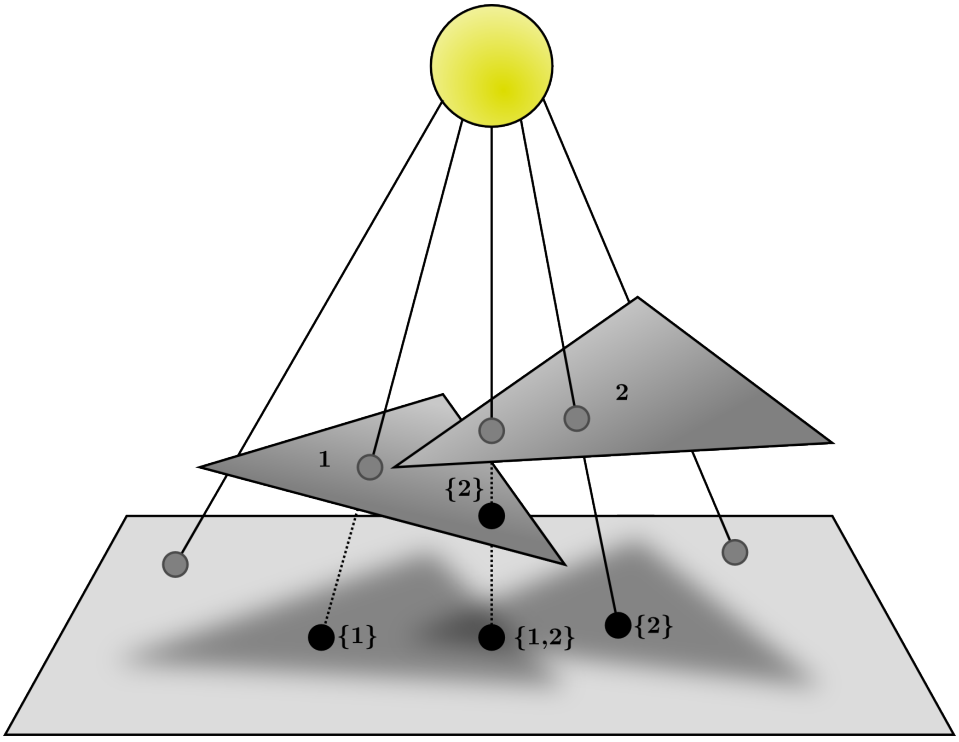
Tijdens het berekenen van de directe belichting van een punt x zoeken we de n dichtstbijzijnde photonen rond het punt x. Op basis van de gevonden photonen beslissen we of we schaduwstralen moeten volgen naar de lichtbronnen. We onderscheiden de volgende situaties op basis van de verzamelde photonen:

- **enkel licht photonen**: we veronderstellen dat er geen enkel blokkend primitief is tussen het punt x en de lichtbron. We gaan er dus vanuit dat de visibiliteit $V(x, y)$ altijd 1 is.
- **enkel schaduw photonen**: we veronderstellen dat we volledig in schaduw zitten. De directe belichting in het punt x is bijgevolg nul.
- **licht en schaduw photonen**: we veronderstellen dat we in een zacht schaduwgebied zitten. Enkel in deze situatie is het noodzakelijk om de visibiliteit te evalueren met schaduwstralen.

3.3 De occlusion map

In deze sectie breiden we de photon map met de schaduw photonen uit tot een datastructuur die toelaat om een schatting van al de kandidaat blokkers rond een punt x te vinden. Hiervoor onthouden we in elk van de schaduw photonen de lijst van blokkende primitieven. Een schaduw photon dat een lijst met blokkende primitieven bevat, noemen we een *occlusion photon*.

De lijst van blokkende primitieven wordt opgebouwd bij het volgen van een straal voor een licht photon. Wanneer we een licht photon aanmaken, slaan we het gesneden oppervlak op in een lijst en volgen we de straal verder doorheen het oppervlak. Bij elke volgende intersectie creëren we een nieuw occlusion photon dat de huidige lijst met blokkers bevat. De lijst wordt daarna uitgebreid met het oppervlak waarop de occlusion photon gecreëerd is. De constructie van de occlusion map is geïllustreerd in figuur 3.7.



Figuur 3.7: Opbouw van de occlusion map in een scene met twee geometrische primitieven 1 en 2. Voor elk van de occlusion photonen staat de lijst met blokkers tussen accolades afgebeeld.

Om een schatting te krijgen van de kandidaat blokkers rond een punt x, zoeken we de n dichtstbijzijnde photonen in de occlusion map. Wanneer we hierbij occlusion photonen vinden, verzamelen we al de potentiële blokkers uit de occlusion photonen in een lijst. Deze lijst geeft ons een benadering van de blokkende primitieven in het punt x.

3.4 Besluit

In dit hoofdstuk beschreven we een datastructuur die ons toelaat om een benadering te vinden van de potentiële blokkers tussen een punt x en een lichtbron. Deze datastructuur, genaamd de *occlusion map*, is een uitbreiding op de photon map met schaduw photonen.

Alhoewel de photon map gebruikt wordt om de globale belichting te berekenen, kan deze ook gebruikt worden om de directe belichting te versnellen. Hiervoor breiden we de photon map uit met schaduw photonen die de gebieden aanduiden die volledig in schaduw liggen. Dankzij de schaduw photonen hoeven we de visibiliteit enkel nog te testen wanneer we rond een punt x zowel schaduw als licht photonen vinden. Over het algemeen kunnen we hierdoor het aantal schaduwstralen reduceren tot 80% [5].

Finaal hebben we de schaduw photonen uitgebreid tot *occlusion photonen* door in elk occlusion photon de gesneden primitieven op te slaan. Door de gesneden primitieven uit de dichtstbijzijnde occlusion photonen rond een punt x te verzamelen, bekomen we een schatting van de kandidaat blokkers van het punt x.

Hoofdstuk 4

Stochastische visibiliteit in een praktisch belichtingsalgoritme

In dit hoofdstuk zullen we de theorie van hoofdstuk 2 integreren in een praktisch belichtingsalgoritme. Hiervoor zullen we de occlusion map uit hoofdstuk 3 gebruiken om de kandidaat blokkers tussen een punt en de lichtbron te vinden.

In de eerste sectie beschrijven we het algoritme om de belichting te berekenen met behulp van stochastische visibiliteit. In de tweede sectie bespreken we de verschillende parameters voor het algoritme en hoe we zinnige waarden voor deze parameters kunnen kiezen. De derde sectie toont de resultaten van het algoritme op drie testscenes. Finaal bespreken we de resultaten en de eigenschappen van het algoritme.

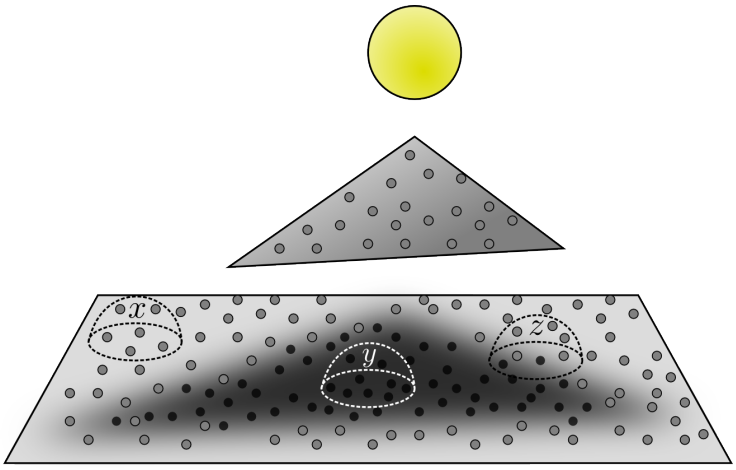
4.1 Integratie van stochastische visibiliteit met de occlusion map

In hoofdstuk 2 hebben we uiteengezet hoe we het product van visibiliteiten kunnen opsplitsen in een som. De visibiliteit wordt geschat door stochastisch één van de termen van de som te evalueren. Om de visibiliteit op te splitsen moeten we echter voor elk punt in de scene de kandidaat blokkers kennen. Om deze te vinden introduceerden we de occlusion map in hoofdstuk 3.

Om de belichting te evalueren gaan we als volgt te werk:

1. We construeren de occlusion map zoals beschreven in sectie 3.3 voor we beginnen met de evaluatie van de belichting.

2. We evalueren de belichting voor elke pixel door een zichtstraal vanuit de camera doorheen de scene te volgen. Rond het eerste intersectiepunt x van deze zichtstraal zoeken we, binnen een vaste straal r, de 100 dichtstbijzijnde photonen in de occlusion map. Hoe de evaluatie van de belichting verder gaat, hangt af van de gevonden photonen. De drie mogelijke situates zijn afgebeeld in figuur 4.1 op de posities x, y en z.



Figuur 4.1: De drie mogelijke combinaties van photonen. In het punt x vinden we enkel licht photonen waardoor we er van uit kunnen gaan dat het punt x volledig belicht is. In het punt y vinden we enkel occlusion photonen waardoor we er vanuit kunnen gaan dat het punt y volledig in schaduw ligt. Op de positie z vinden we een mix van licht en occlusion photonen.

- In het punt x vinden we geen enkel occlusion photon. Hierbij gaan we er van uit dat het punt x volledig belicht is. Bijgevolg hoeven we geen schaduwstralen te volgen om de visibiliteit te evalueren.
- In het punt y vinden we enkel occlusion photonen. Hierbij gaan we er vanuit dat het punt x volledig in schaduw ligt. De belichting in het punt y is nul en in deze situatie hoeven we dus ook geen schaduwstralen te volgen.
- In het punt z vinden we zowel occlusion als licht photonen. We bevinden ons in een zachte schaduw. Enkel in deze situatie is het nodig om de visibiliteit te evalueren.

We hoeven de visibiliteit dus enkel te evalueren wanneer we zowel occlusion photonen als licht photonen vinden rond het punt x. We construeren de verzameling $Z = \{z_1, z_2, ..., z_n\}$ van kandidaat blokkers door al de blokkers uit de gevonden occlusion photonen te verzamelen. Deze verzameling splitsen we in twee gelijke groepen A en B zoals besproken in sectie 2.2.2 zodat:

$$V_A(x,y) = \prod_{i=1}^{n/2-1} V_{z_i}(x,y) \tag{4.1}$$

$$V_B(x,y) = \prod_{i=n/2}^{n} V_{z_i}(x,y) \tag{4.2}$$

Met deze twee groepen kunnen we de visibiliteit stochastisch evalueren met behulp van formule 2.6, 2.40 of 2.47.

4.2 Parameters voor het algoritme

Het algoritme beschreven in sectie 4.1 bevat veel parameters die we kunnen aanpassen om het algoritme te optimaliseren. Deze parameters zijn:

- $n_{photonen}$: het aantal photonen dat we doorheen de scene distribueren.
- n_{zoek}: het maximale aantal photonen dat we proberen te verzamelen wanneer we de dichtstbijzijnde photonen rond een punt x zoeken.
- r: de maximale afstand van een photon ten opzichte van het punt x waarrond we de photonen zoeken.
- de stochastische visibiliteitsformule waarmee we de visibiliteit evalueren.
- p_1, p_2 en p_3; de kansen voor elk van de termen van de stochastische visibiliteitsformule.
- het verdelen van de blokkers over de twee groepen.

Het is onmogelijk om al deze parameters exhaustief uit te proberen. Bovendien variëren de optimale parameters van scene tot scene. We kunnen echter enkele zinvolle benaderingen maken.

Als A_{scene} de oppervlakte is van de hele geometrie in de scene en als we er van uit gaan dat de photonen uniform verdeeld zijn over de geometrie, dan is de dichtheid aan photonen in elk punt van de scene gelijk aan:

$$\rho_{photonen} = \frac{n_{photonen}}{A_{scene}} \tag{4.3}$$

Als we minstens n_{zoek} photonen willen vinden over een oppervlak met grootte πr^2, dan weten we dat de dichtheid aan photonen minimaal gelijk moet zijn aan:

$$\rho_{zoek} = \frac{n_{zoek}}{\pi r^2} \tag{4.4}$$

Hierbij kunnen we dan stellen dat het minimum aantal benodigde photonen gelijk moet zijn aan:

$$\rho_{photonen} \geq \rho_{zoek} \tag{4.5}$$

$$\frac{n_{photonen}}{A_{scene}} \geq \frac{n_{zoek}}{\pi r^2} \tag{4.6}$$

$$n_{photonen} \geq n_{zoek} \frac{A_{scene}}{\pi r^2} \tag{4.7}$$

Vergelijking 4.7 levert ons een intuïtieve formule op. Deze stelt dat het aantal benodigde photonen in de scene groter wordt wanneer:

- we meer photonen willen vinden rond een punt x
- we evenveel photonen willen vinden in een kleinere straal rond x

Bovendien kunnen we vergelijking 4.7 gebruiken om onze parameters bij benadering te kiezen. Bij de traditionele photon map probeert men steeds minstens tussen de 50 en 100 photonen te vinden om de radiantie te schatten. Uit onze experimenten stelden we vast dat $n_{zoek} = 100$ goede resultaten oplevert in de meeste scenes. Daarnaast willen we omwille van performantie en geheugen het aantal photonen in de scene beperken. Hierbij beperken we het maximale aantal photonen tot 3000000. Met deze voorwaarden kunnen we voor iedere scene de zoekradius bepalen:

$$n_{photonen} \geq n_{zoek} \frac{A_{scene}}{\pi r^2} \tag{4.8}$$

$$r \geq \sqrt{\frac{n_{zoek} A_{scene}}{n_{photonen} \pi}} \tag{4.9}$$

$$r \geq \sqrt{\frac{A_{scene}}{30000\pi}} \tag{4.10}$$

4.3 Resultaten

In deze sectie passen we het algoritme van sectie 4.1 toe op drie testscenes. De afbeeldingen van de testscenes zijn gerenderd met 4 monsters per pixel waarbij we de lichtbron bemonsteren met 128 schaduwstralen. We gebruiken de benaderingen uit 4.2 om de parameters voor het algoritme te bepalen. Deze parameters zijn opgesomd in tabel 4.1. Daarnaast zullen we de kansen p_1, p_2 en p_3 gelijkstellen aan 1/3. Voorlopig zullen we ook voor elke scene de kandidaat blokkers willekeurig in twee groepen verdelen. De overige parameters voor de testscenes zijn gegeven in tabel 4.1

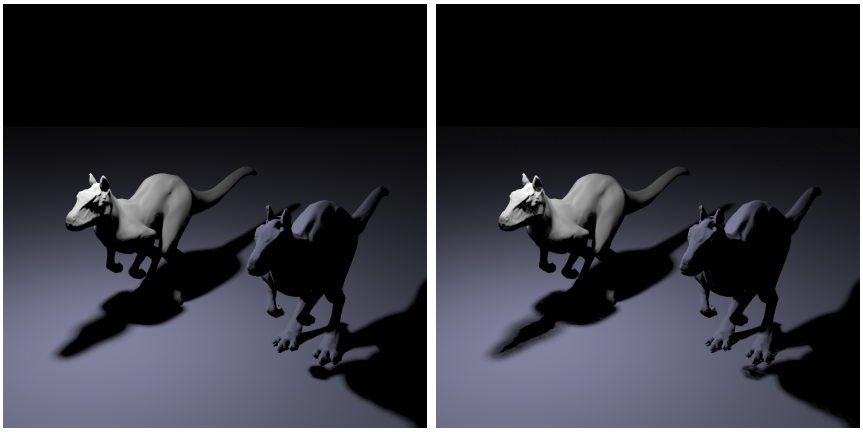
Scene	A_{scene}	$n_{photonen}$	n_{zoek}	r
Killeroos	1033580	3000000	100	3.3
Cornell met Draak en Killeroo	1864116	3000000	100	4.5
Cornell met icosahedronen	1755910	3000000	100	4.31

Tabel 4.1: Parameters voor de verschillende testscenes.

4.3.1 Opzet

Om objectief de invloed van stochastische visibiliteit te schatten, vergelijken we stochastische visibiliteit met een algoritme dat de occlusion map gebruikt, maar de visibiliteit niet zal opsplitsen. Dit algoritme zullen we het *occlusion map* algoritme noemen. Het occlusion map algoritme is analoog met het stochastische algoritme uit sectie 4.1, alleen zal dit algoritme de visibiliteit deterministisch testen.

4.3.2 Killeroos scene



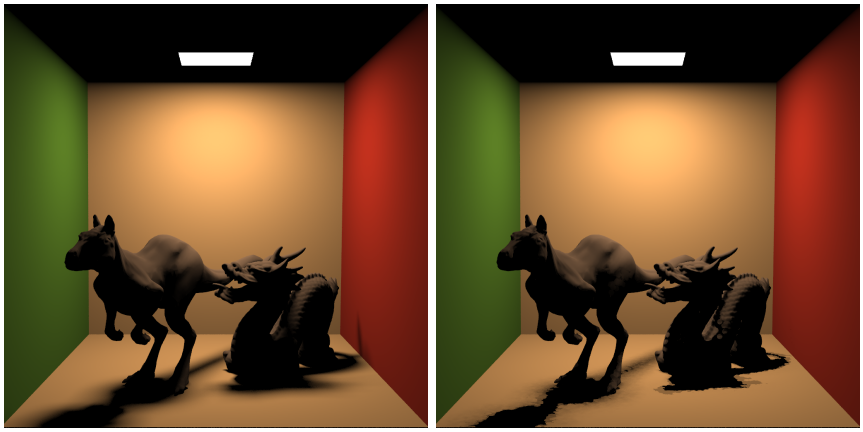
FIGUUR 4.2: Deterministische evaluatie voor de Killeroos scene. De linker afbeelding is gerenderd met deterministische visibiliteit. De rechter afbeelding is gerenderd met het occlusion mapping algoritme door deterministisch al de blokkers gevonden in de occlusion map te testen.

FIGUUR 4.3: Stochastische evaluatie voor de Killeroos scene. De linker afbeelding is gerenderd met merkwaardig product formule #1, de middelste afbeelding is gerenderd met merkwaardig product formule #2, de rechter afbeelding is gerenderd met de binomiaal formule.

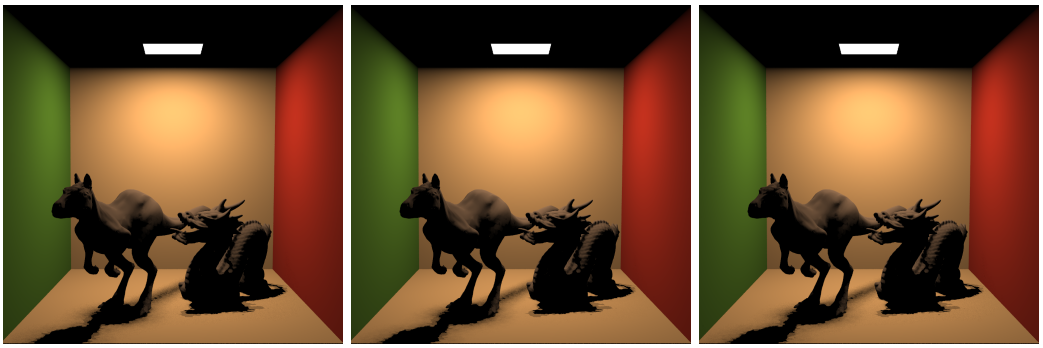
algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (×1000000)
Stochastische visibiliteit	merkwaardig product #1	59.2	1375
	merkwaardig product #2	63.7	1610
	binomiaal	63.5	1610
Occlusion map	-	67.8	1957
Exacte visibiliteit	-	72.4	1141

TABEL 4.2: Resultaten van de Killeroos scene.

4.3.3 Draak met Killeroo



Figuur 4.4: Deterministische evaluatie voor de Draak met Killeroo scene. De linker afbeelding is gerenderd met deterministische visibiliteit. De rechter afbeelding is gerenderd met het occlusion mapping algoritme door deterministisch al de blokkers gevonden in de occlusion map te testen.

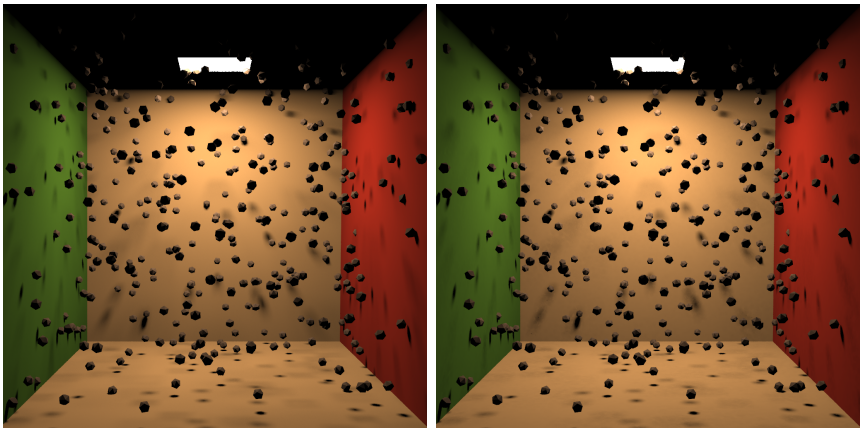


Figuur 4.5: Stochastische evaluatie voor de Cornell box met Draak en Killeroo. De linker afbeelding is gerenderd met merkwaardig product formule #1, de middelste afbeelding is gerenderd met merkwaardig product formule #2, de rechter afbeelding is gerenderd met de binomiaal formule.

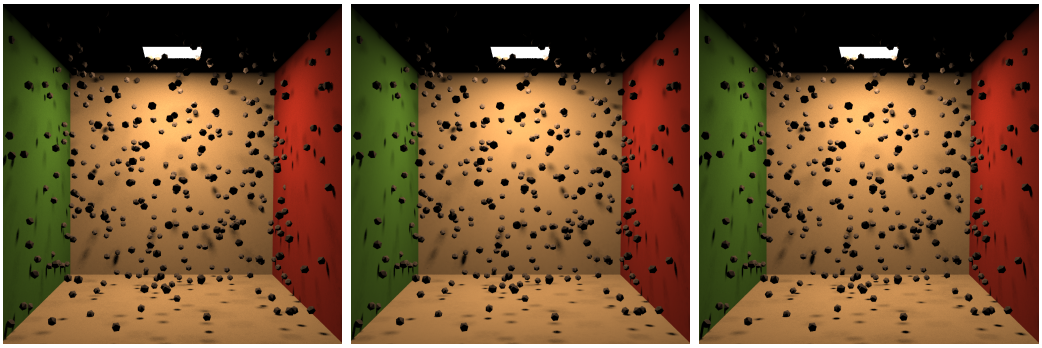
algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (×1000000)
Stochastische visibiliteit	merkwaardig product #1	58.7	724
	merkwaardig product #2	58.5	883
	binomiaal	58.4	885
Occlusion map	-	61.8	1194
Exacte visibiliteit	-	108.2	1489

Tabel 4.3: Resultaten van de Draak met Killeroo scene.

4.3.4 Cornell box met icosahedronen



FIGUUR 4.6: Deterministische evaluatie voor de Cornell box met icosahedronen. De linker afbeelding is gerenderd met deterministische visibiliteit. De rechter afbeelding is gerenderd met het occlusion mapping algoritme door deterministisch al de blokkers gevonden in de occlusion map te testen.



FIGUUR 4.7: Stochastische evaluaties voor de Cornell box met icosahedronen. De linker afbeelding is gerenderd met product formule #1, de middelste afbeelding is gerenderd met product formule #2, de rechter afbeelding is gerenderd met de binomiaal formule.

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (×1000000)
Stochastische visibiliteit	merkwaardig product #1	118	2638
	merkwaardig product #2	122	3079
	binomiaal	126	3074
Occlusion map	-	133	4028
Exacte visibiliteit	-	122	1495

TABEL 4.4: Resultaten van de Cornell box met icosahedronen.

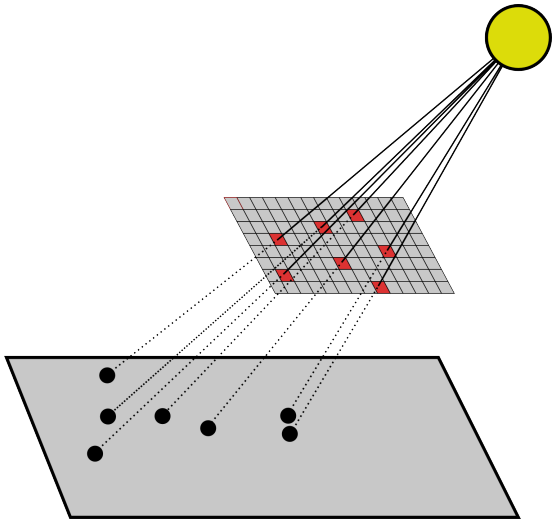
4.4 Evaluatie van het algoritme

4.4.1 Convergentie

Dankzij de occlusion map kunnen we in een scene onderscheid maken tussen de volledig belichte, de volledig beschaduwde en de zachte schaduw gebieden in de scene. Omdat we in de volledig belichte en volledig beschaduwde gebieden geen schaduwstralen hoeven te volgen zal er in deze gebieden geen ruis optreden door de stochastische visibiliteit.

We zien echter uit de figuren dat het algoritme niet convergeert naar het juiste resultaat. De oorzaak ligt echter bij de occlusion map en niet bij de stochastische visibiliteit. Dit blijkt uit figuren 4.2, 4.4 en 4.6. De linker helft van elk van deze figuren toont de scene gerenderd met exacte visibiliteit. De rechter helft is gerenderd met het occlusion map algoritme. Het occlusion map algoritme test deterministisch al de kandidaat blokkers gevonden rond een punt x. Hierdoor toont de rechter afbeelding steeds het beste resultaat dat we kunnen bereiken met stochastische visibiliteit.

De occlusion map slaagt er niet in om al de potentiële blokkers op te slaan. Tijdens het verdelen van de photonen bestaat de kans dat de fijne geometrie in de scene nooit gesneden zal worden. Die fijne geometrie zal dus niet opgenomen zijn in de occlusion map, waardoor die geometrie nooit visibiliteitsevents kan genereren. Deze situatie is afgebeeld in figuur 4.8. Indien de scene bestaat uit zeer fijne geometrie, dan is de kans groot dat een deel van de geometrie gemist wordt. Een natuurlijke oplossing is meer photonen in de scene verdelen, zoals afgebeeld in figuur 4.9. Dit gaat wel ten koste van performantie en geheugen. In hoofdstuk 5 zullen we een robuustere oplossing voorstellen.



Figuur 4.8: Scene met een vlak dat bestaat uit zeer fijne geometrie. Indien we slechts een beperkt aantal photonen distribueren zal slechts een klein deel van deze fijne geometrie opgenomen worden in de occlusion map.

4.4.2 Performantie

Intersectietesten

Uit tabellen 2.1, 2.2 en 2.3 zien we dat we met behulp van stochastische visibiliteit altijd intersectietesten winnen ten opzichte van de occlusion map. Dit is een logisch gevolg vermits we met de occlusion map de hele set van potentiële blokkers moeten testen, terwijl we met stochastische visibiliteit deze set opdelen in twee kleinere groepen. Uit de theorie van hoofdstuk 2 voorspelden we dat we ongeveer 33% aan intersectie testen besparen. Tabel 4.5 toont hoeveel intersectie testen we besparen ten opzichte van de occlusion map.

scene	**intersectie-testen** occlusionmap	**intersectie-testen** product #1	**intersectie-testen** product #2	**intersectie-testen** binomiaal
Killeroos	1957	1375 (-30%)	1610 (-17%)	1610 (-17%)
Cornell box met Draak en Killeroo	1194	724 (-40%)	885 (-26%)	883 (-26%)
Cornell box met icosahedronen	4028	2638 (-34%)	3074 (-24%)	3079 (-27%)

TABEL 4.5: Invloed van stochastische visibiliteit op het aantal intersectietesten.

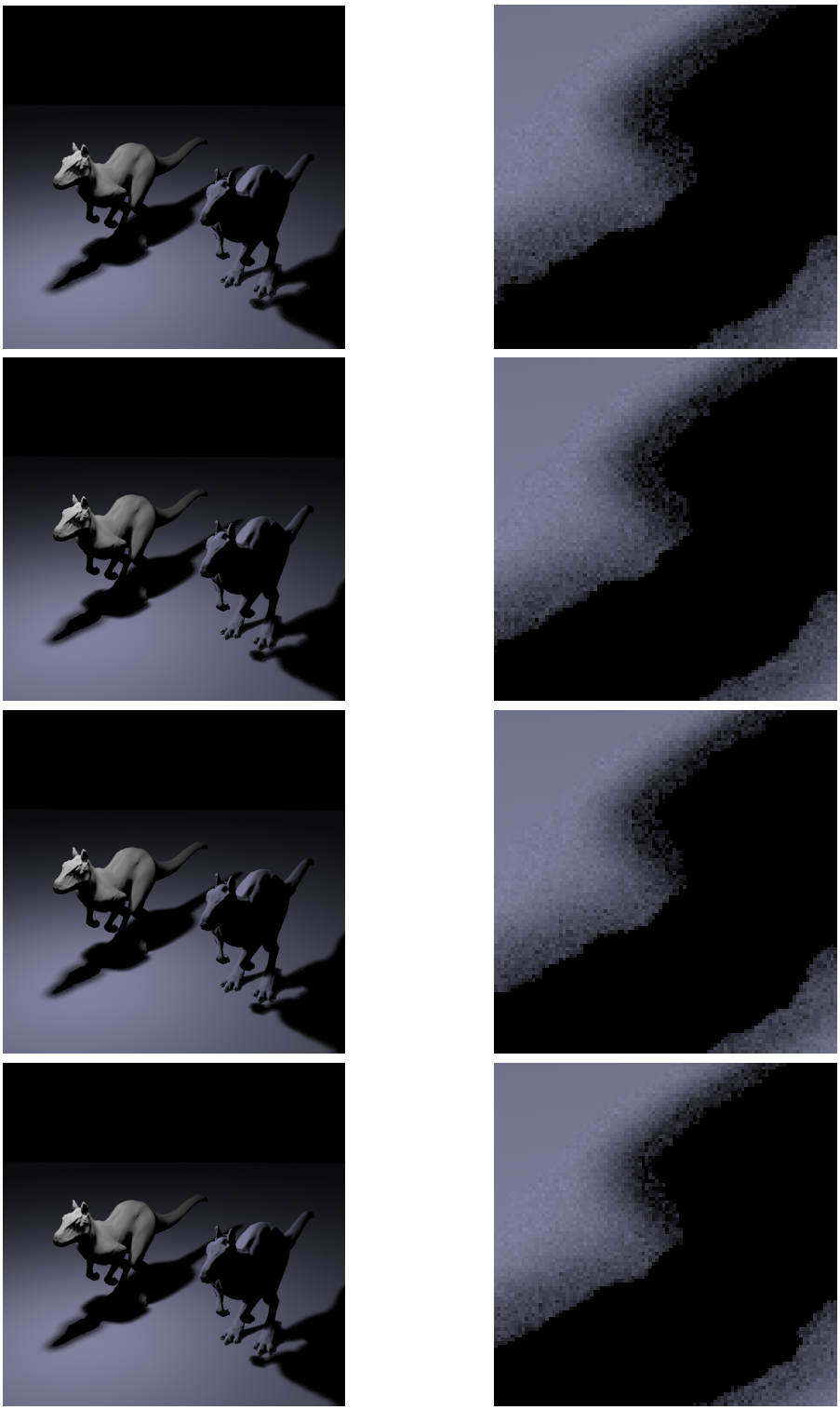
Uit tabel 4.5 zien we dat we procentueel gemiddeld 26% intersecties uitsparen. Wanneer we de visibiliteitsformules met elkaar vergelijken, dan zien we dat we minder intersecties moeten uitvoeren wanneer we de visibiliteit opsplitsen met behulp van merkwaardig product #1. Dit kunnen we verklaren door het complementaire visibiliteitsproduct $\overline{V_A(x,y)} \cdot \overline{V_B(x,y)}$ in formule 2.34. Wanneer $\overline{V_A(x,y)}$ gelijk is aan 0, dan is het product gelijk aan 0 ongeacht de waarde van $\overline{V_B(x,y)}$, waardoor we een groep minder hoeven te intersecteren.

Geheugengebruik

Om het geheugengebruik te schatten, schetsen we eerst hoe de photonen worden opgeslagen. Elk photon slaat zijn positie en richting op met behulp van zes floating point getallen. Bovendien slaat elk licht photon ook de flux van de lichtbron op met behulp van drie floating point getallen. Daarnaast onthoudt elk occlusion photon zijn lijst met kandidaat blokkers.

Elk licht photon neemt dus 36 bytes aan geheugen in beslag. Een occlusion photon neemt 24 bytes in beslag plus 4 extra bytes voor elke kandidaat blokker die opgeslagen moet worden.

Omdat de photonen in een kd-boom gegroepeerd worden, moeten we ook het geheugengebruik van de kd-bomen in rekening brengen. Als we voor N photonen een gebalanceerde kd-boom construeren, dan heeft deze boom een hoogte $h = \log_2(N)$. Het aantal knopen K in een boom van hoogte h is gelijk aan $2^{h+1} - 1$. Dit kunnen



Figuur 4.9: Convergentie bij het verdelen van meer photonen doorheen de ruimte. De rijen zijn telkens gerenderd met 6, 9, 12 en 16 miljoen photonen waarbij de zoekradius r aangepast is met behulp van formule 4.7.

we vereenvoudigen tot:

$$K = 2^{h+1} - 1 = 2 \cdot 2^{\log_2(N)} - 1 = 2N - 1 \tag{4.11}$$

Elke knoop in de kd-boom onthoudt de positie en de as waarop het vlak geplaatst is. Hiervoor zijn 8 bytes nodig. Daarnaast heeft elke knoop een referentie naar zijn twee kindknopen, wat ook 8 bytes vereist. In totaal heeft elke knoop dus 16 bytes geheugen nodig.

We kunnen nu een gesloten formule voor het geheugengebruik in bytes geven voor N_{licht} licht photonen en voor $N_{occlusion}$ occlusion photonen die gemiddeld d kandidaat blokkers bevatten:

$$\begin{aligned} M = &(2N_{licht} - 1) \cdot 16 \\ &+ (2N_{occlusion} - 1) \cdot 16 \\ &+ N_{licht} \cdot 36 + N_{occlusion} \cdot (24 + d \cdot 4) \end{aligned} \tag{4.12}$$

In onze experimenten is het gemiddeld geheugengebruik gelijk aan:

scene	$\mathbf{N_{occlusion}}$	$\mathbf{N_{licht}}$	**d**	**M**
Killeroos	509845	2499050	1.86	192 Mib
Cornell box met Draak en Killeroo	361464	2653822	2.04	194 Mib
Cornell box met icosahedronen	713952	2313292	1.55	192 Mib

TABEL 4.6: Geheugengebruik van de occlusion map voor de verschillende testscenes.

4.5 Besluit

In dit hoofdstuk hebben we een algoritme uiteengezet dat met behulp van de occlusion map en stochastische visibiliteit de directe belichting in een scene kan berekenen. Hierbij zagen we echter dat de occlusion map in zijn huidige vorm niet voldoet om goede resultaten te krijgen. Indien de scene fijne geometrie bevat, dan is de kans groot dat deze fijne geometrie gemist wordt tijdens de constructie van de occlusion map. De gemiste geometrie is dus niet opgenomen in de occlusion map en zal dus nooit getest worden.

De stochastische visibiliteit slaagde er echter wel in om het aantal intersectie testen te reduceren met gemiddeld 20%. Het percentage lag gemiddeld lager bij de formule op basis van product #1 omdat we hier de visibiliteitsevaluatie van het product met de complementaire visibiliteiten vroegtijdig kunnen afbreken indien een van de termen gelijk is aan 0.

Hoofdstuk 5

Uitbreidingen

In hoofdstuk 4 integreerden we stochastische visibiliteit met de occlusion map tot een praktisch algoritme om de belichting te berekenen. De occlusion map schiet echter tekort vermits de fijne geometrie in de scene vaak gemist wordt. Hierdoor convergeert het algoritme met stochastische visibiliteit niet naar het juiste resultaat.

In dit hoofdstuk beschrijven we enkele verbeteringen waardoor het algoritme performanter wordt.

5.1 Volumetrische occluders

Uit het vorige hoofdstuk bleek dat de occlusion map er niet in slaagde om de fijne geometrie in de scene op te slaan. Wanneer de geometrie in de scene klein is, dan is de kans groot dat deze fijne geometrie gemist wordt. De kans dat een willekeurige straal $r(x, \Theta)$ vanuit de lichtbron een oppervlak A_i raakt is evenredig met de ruimtehoek van het oppervlak A_i ten opzichte van de oppervlakte van de hemisfeer rond het punt x:

$$p_{raak} = \frac{\Omega_{A_i}}{2\pi} = \frac{A_i}{2\pi r^2} \tag{5.1}$$

Uit deze formule blijkt dus dat de raakkans kleiner wordt naarmate het oppervlak A_i kleiner is en naarmate het oppervlak A_i verder van het punt x ligt.

In het vorige hoofdstuk losten we het probleem van de gemiste geometrie op door meer photonen doorheen de ruimte te verspreiden. Dit heeft echter een negatief effect op de performantie en het geheugengebruik. In deze sectie lossen we dit probleem op door meer kandidaat blokkers te introduceren.

Om meer kandidaat blokkers te creëren, vullen we de waterdichte modellen in de scene met volumetrische occluders [2]. Het idee is dat deze nieuwe occluders een veel grotere oppervlakte hebben en hierbij compenseren voor het missen van de fijne geometrie.

Het construeren van de volumetrische occluders gebeurt door een octree structuur te bouwen over de waterdichte modellen. We berekenen eerst de omhullende kubus die het waterdichte model compleet bevat. Deze omhullende kubus delen we recursief op in vier gelijke kubussen.

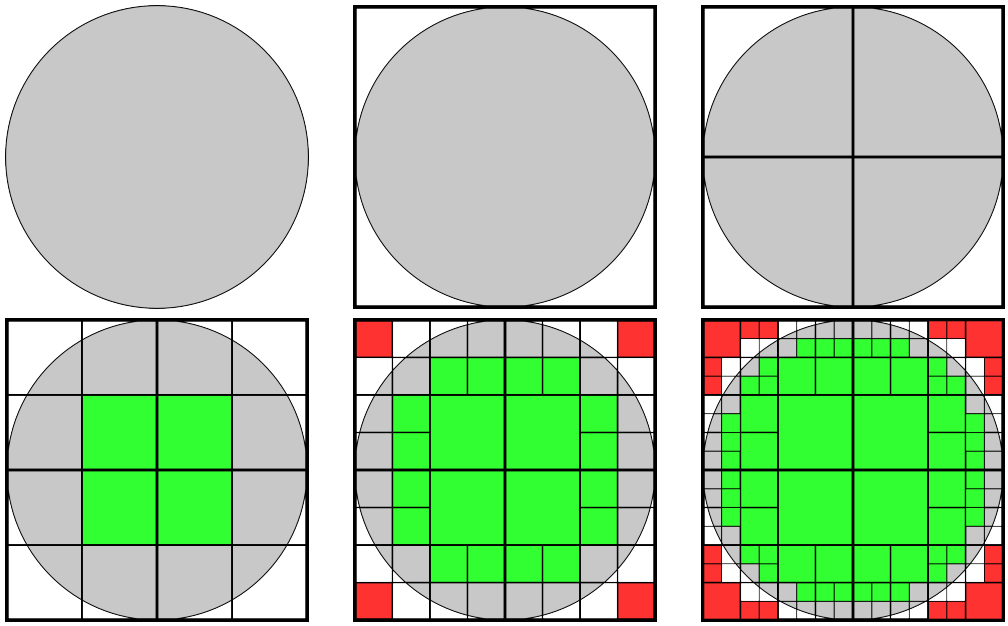
We stoppen de recursieve opdeling wanneer een kubus geen geometrie meer bevat of wanneer we een maximale recursiediepte bereiken.

Voor elke lege kubus moeten we testen of de kubus binnen of buiten het waterdichte model ligt. Omdat het model waterdicht is, volstaat het om een willekeurige schaduwstraal te schieten vanuit de lege kubus. Hierbij onderscheiden we twee mogelijke situaties:

- **de kubus ligt buiten het model**: in dit geval mist de schaduwstraal het waterdichte model, ofwel raakt de schaduwstraal het model, maar ligt de normaal van het geraakte oppervlak in de tegenovergestelde richting van de richting van de schaduwstraal.

- **de kubus ligt binnen het model**: in dit geval moet de schaduwstraal het waterdichte model raken en dan moet de normaal van het geraakte oppervlak in de dezelfde richting liggen als de richting van de schaduwstraal.

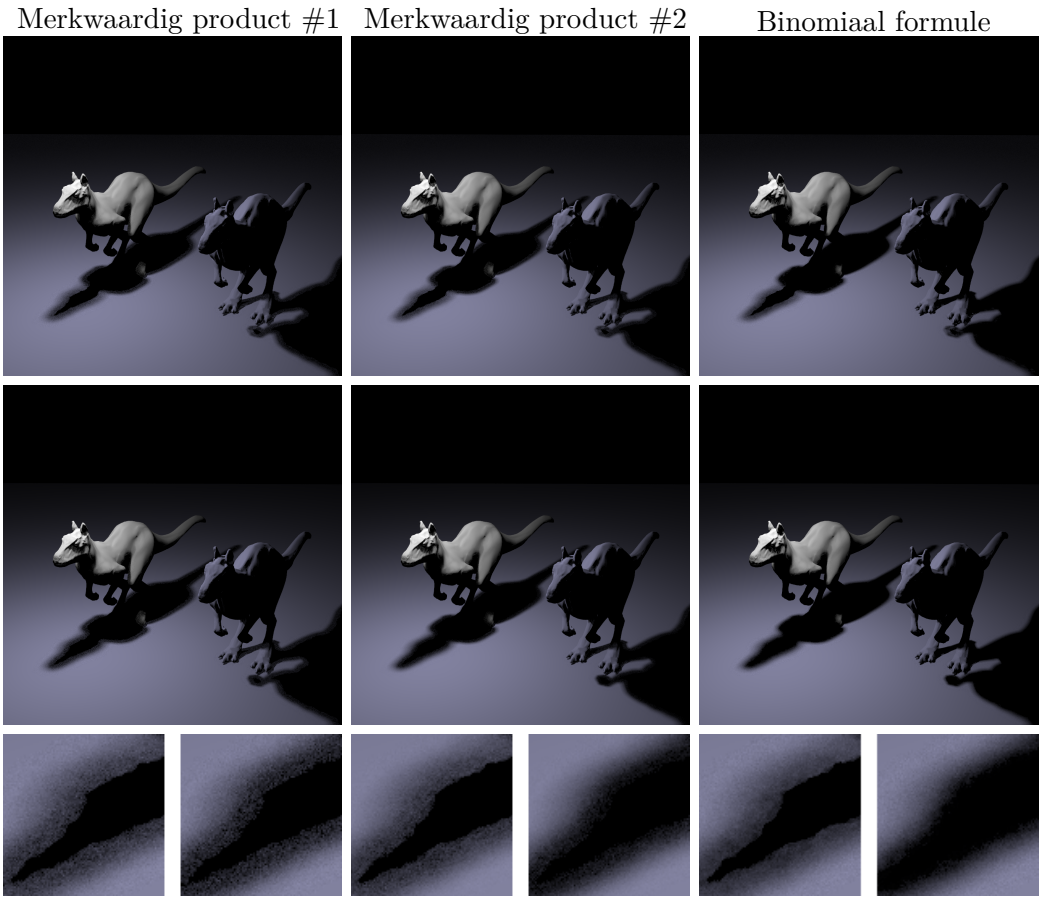
De kubussen die binnen het model liggen, worden daarna gebruikt als volumetrische occluders en opgenomen als geometrie in de scene. Het opbouwen van de volumetrische occluders wordt getoond in figuur 5.1.



Figuur 5.1: Constructie van de volumetrische occluders in 2D voor een cirkel. Hierbij zoeken we eerst de omhullende rechthoek van de cirkel. Deze rechthoek delen we recursief op tot een maximale diepte van 5. Indien een rechthoek geen rand van de cirkel bevat testen we of de rechthoek binnen of buiten de cirkel ligt. De groene rechthoeken liggen binnen de cirkel en zullen als volumetrische occluders gebruikt worden. De rode rechthoeken liggen buiten de cirkel. De overige rechthoeken bevatten nog steeds de rand van de cirkel.

Deze volumetrische occluders worden gecreëerd voordat we de occlusion map voor de scene maken. De volumetrische occluders worden toegevoegd als geometrie in de scene en hierdoor ook opgenomen in de occlusion map.

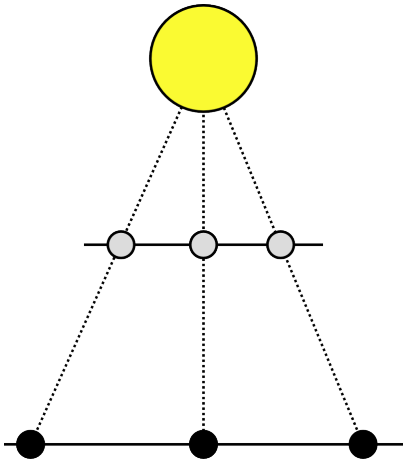
Figuur 5.2 toont het effect van volumetrische occluders in de Killeroos scene.



FIGUUR 5.2: Vergelijking van stochastische visibiliteit met en zonder volumetrische occluders. De figuren zijn van links naar rechts gerenderd met respectievelijk merkwaardig product #1, merkwaardig product #2 en met de binomiaal formule. De eerste rij toont de killeroo scene zonder volumetrische occluders. De tweede rij toont de killeroo scene met volumetrische occluders. De derde rij toont close-ups met links de close-up zonder volumetrische occluders en rechts de close-up met volumetrische occluders.

5.2 Distributie van de photonen

In hoofdstuk 4 gebruiken we de mechanismen van de photon map met schaduw photonen om de licht en occlusion photonen doorheen de ruimte te verspreiden. Hierbij werden de photonen verspreid vanaf de lichtbron [5]. Het nadeel van deze aanpak is dat de distributie van de photonen niet uniform is over de ruimte. De concentratie aan photonen is groter op oppervlakken dichter bij de lichtbron dan op oppervlakken verder van de lichtbron, zoals geïllustreerd in figuur 5.3. Daarom stellen we in deze sectie enkele andere mogelijkheden voor om de photonen te distribueren in de scene.



Figuur 5.3: Wanneer de photonen vanaf de lichtbron gedistribueerd worden, vermindert de densiteit van de photonen met de afstand tot de lichtbron.

5.2.1 Uniforme distributie

Om de photonen gelijkmatiger te verdelen door de scene gaan we de locaties van de photonen uniform kiezen over de oppervlakken in de scene. Om dit te doen kiezen we willekeurig een punt x op een oppervlak A_i in de scene. De kans dat we het oppervlak A_i kiezen is evenredig met de totale oppervlakte A_{totaal} van de geometrie in de scene.

$$p(x) \simeq \frac{A_i}{A_{totaal}} \tag{5.2}$$

Vanuit het punt x volgen we een schaduwstraal naar een willekeurig punt y op de lichtbron. Indien deze schaduwstraal geometrische primitieven raakt, creëren we een occlusion photon dat al de geraakte geometrische primitieven bevat. Wanneer de schaduwstraal ongehinderd de lichtbron raakt creëren we een licht photon.

In tegenstelling tot het verdelen van de photonen vanaf de lichtbron garandeert deze techniek dat de photonen uniform doorheen heel de ruimte verdeeld zijn.

5.2.2 Camera distributie

De uniforme distributie verhelpt het probleem dat de photonen niet gelijkmatig doorheen de ruimte verdeeld zijn. Een nadeel van zowel de uniforme distributie als de distributie vanaf de lichtbron is dat de photonen vaak terecht komen in delen van de scene die niet zichtbaar zijn vanuit de virtuele camera.

Om de photonen beter te verdelen in de zichtbare regio van de scene, genereren we willekeurige zichtstralen vanuit de camera. We gebruiken het eerste intersectie punt x van de zichtstraal als positie om een photon te creëren. Analoog aan de uniforme distributie volgen we een schaduwstraal naar een willekeurig punt y op de lichtbron. Indien deze schaduwstraal geen enkel geometrisch primitief raakt, creëren we een licht photon. Anders creëren we een occlusion photon dat al de gesneden geometrische primitieven tussen het punt x en y bevat.

Met behulp van de camera distributie kunnen we de photonen verdelen in de relevante delen van de scene. Het nadeel is echter dat we de occlusion map niet kunnen hergebruiken wanneer we de scene renderen vanuit een ander camera standpunt.

5.2.3 Adaptieve camera distributie

Uit het algoritme van hoofdstuk 4 weten we dat we de visibiliteit enkel evalueren wanneer we een mix vinden van licht photonen en occlusion photonen rond het punt x dat we willen belichten. Om een goede benadering te krijgen van de kandidaat blokkers willen we zoveel mogelijk occlusion photonen distribueren in de zachte schaduwgebieden van de scene. Hiervoor hergebruiken we de camera distributie uit sectie 5.2.2 waarbij we meer zichtstralen gaan volgen naar de zachte schaduwgebieden van de scene.

We gebruiken hiervoor een adaptief bemonsteringschema voor de pixels. We stratificeren de pixel ruimte in strata van 8x8 pixels. Vervolgens distribueren we een aantal test photonen doorheen de ruimte analoog aan de standaard camera distributie. Hierbij houden we bij hoeveel occlusion photonen en licht photonen er in elke strata gecreëerd zijn. Met behulp van deze data creëren we een kansdistributiefunctie voor elk stratum zodat strata met een hogere concentratie aan licht en occlusion photonen een grotere kans hebben om geselecteerd te worden. Experimenteel hebben we vastgesteld dat de volgende kansdistributie functie goede resultaten geeft:

$$p\left(strata_i\right) \simeq \frac{n_{occlusion}}{n_{strata}} + 0.75 n_{occlusion_i} * \left(0.1 + {n_{licht_i}}^{0.3}\right) \tag{5.3}$$

Hierbij staat $n_{occlusion}$ voor het totale aantal gedistribueerde occlusion photonen en staan $n_{occlusion_i}$ en n_{licht_i} respectievelijk voor het aantal occlusion en licht photonen gecreëerd in statum i.

Met behulp van deze kansdistributiefunctie kiezen we de zichtstralen om de photonen de distribueren. Eerst selecteren we een stratum volgens de kansdistributiefunctie 5.3. Vervolgens selecteren we een willekeurige pixel binnen dit stratum om een zichtstraal door te schieten.

Experimenteel vonden we dat 200000 testphotonen voldoende zijn om de kansdistributiefunctie op te stellen. Na elke set van 20000 photonen herevalueren we de kansdistributiefunctie.

5.2.4 Vergelijking van de distributies

In deze sectie analyseren we het effect van de verschillende photon distributie methodes. Figuur 5.4 toont de resulterende afbeeldingen voor de killeroo scene gerenderd met de verschillende photon distributie methodes. Elke afbeelding is gerenderd met de binomiaal formule, gebruik makend van 4 samples per pixel en 128 schaduwstralen. Figuur 5.5 toont enkele close-ups uit de resulterende afbeeldingen. Figuur 5.6 toont het aantal occlusion photonen per pixel in de afbeeldingen. Tabel 5.1 beschrijft de inhoud van de occlusion map voor de verschillende distributies in de killeroo scene.

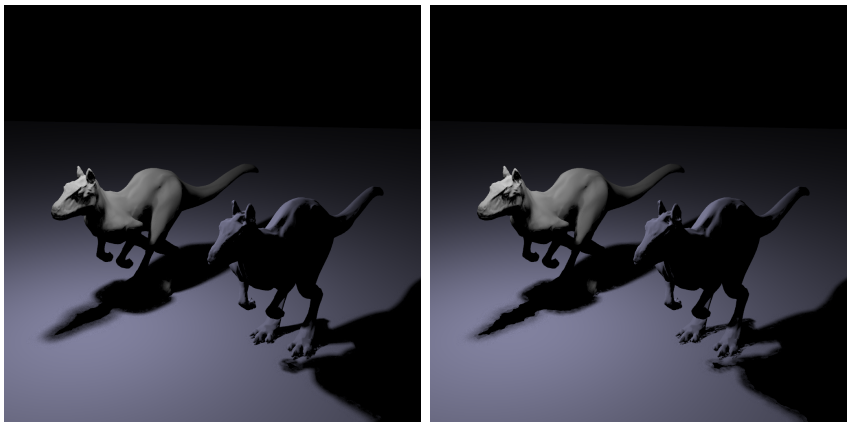
distributie methode	$\mathbf{n_{occlusion}}$	$\mathbf{n_{licht}}$	**d**	**intersecties**
licht	510425	2502104	1.85	9.2M
uniform	132672	2884474	1.94	2.7M
camera	736357	2273302	2.07	13.6M
camera adaptief	1682526	1332646	2.03	34.3M

Tabel 5.1: Data in de occlusion map voor de Killeroos scene. n_{licht} is het aantal licht photonen in de occlusion map, $n_{occlusion}$ is het aantal occlusion photonen in de occlusion map en d is het gemiddeld aantal kandidaat blokkers per occlusion photon. Finaal toont de tabel ook het aantal intersecties benodigd om de occlusion map te creëren.

Uit de tabel en de figuren kunnen we afleiden dat de uniforme distributie in deze scene slechter presteert. Hoewel de uniforme distributie het minste aantal intersecties vraagt om op te bouwen bevat deze het laagste aantal occlusion photonen. Hierdoor zien we in de close-ups van figuur 5.5 dat er nagenoeg geen schaduw is in de gebieden waar schaduw zou moeten zijn.

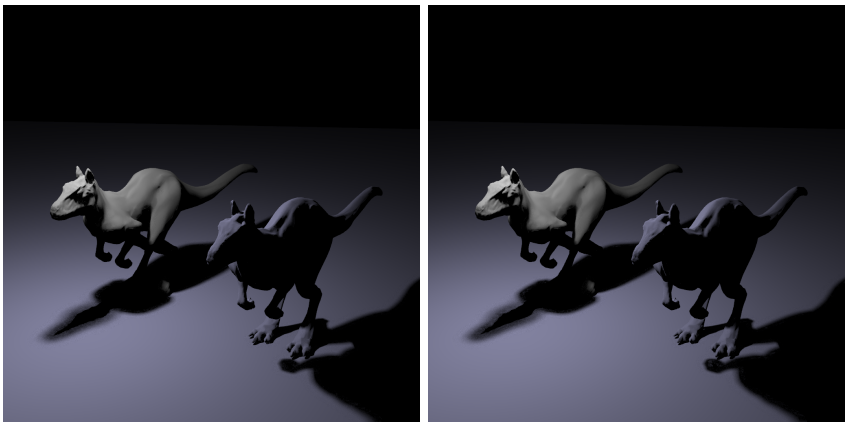
De photon distributie vanuit de camera met en zonder het adaptief schema vereisen een veel groter aantal intersectietesten om de occlusion map te construeren. Dit komt omdat we voor deze distributie methodes ook zichtstralen schieten vanuit de camera. Hier staat wel tegenover dat de densiteit aan occlusion photonen veel groter is. Bovendien worden ze enkel gedistribueerd in de zichtbare gebieden van de scene. Uit de close-ups zien we dan ook dat de schaduwregio's veel donkerder zijn.

Verder zien we uit figuur 5.6 dat de adaptieve camera distributie methode er zeer goed in slaagt om een groot deel van de photonen te verdelen in de zachte schaduwregio van de scene.



Photon distributie vanaf de lichtbron

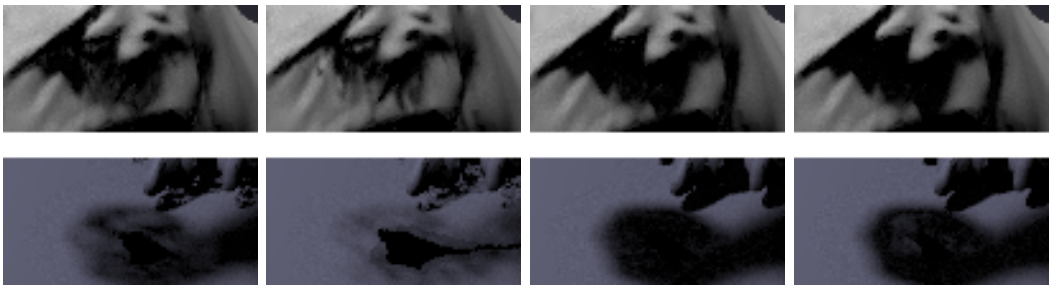
Uniforme photon distributie



Photon distributie vanaf de camera

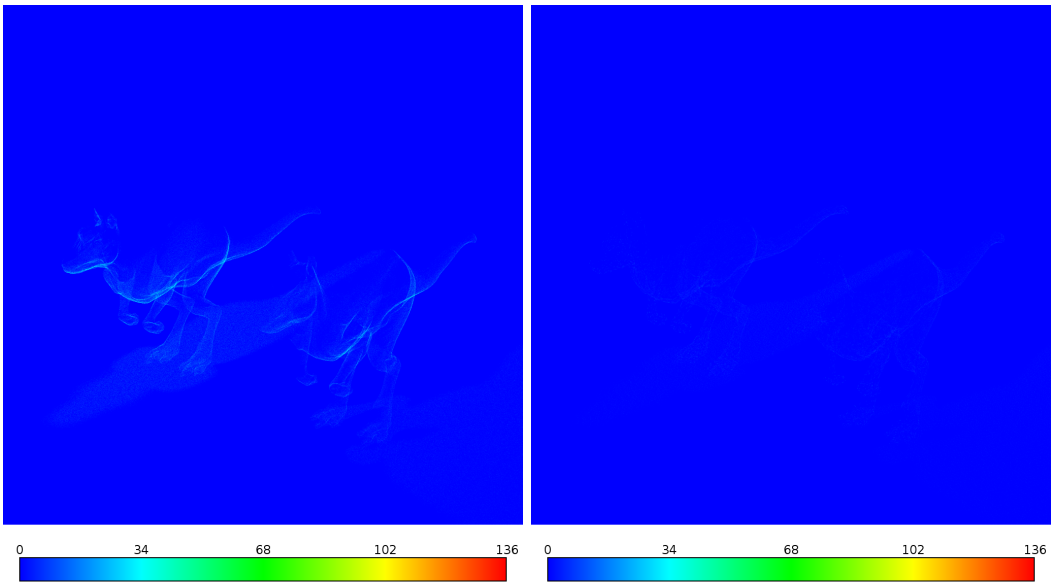
Adaptieve photon distributie vanaf de camera

FIGUUR 5.4: Resulterende afbeeldingen met de verschillende photon distributie methodes



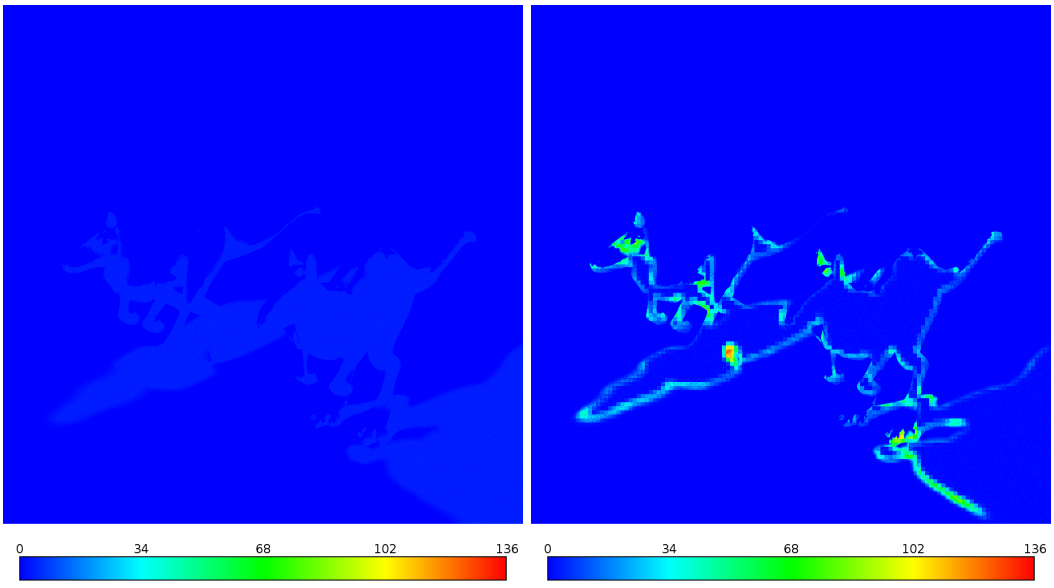
(a) (b) (c) (d)

FIGUUR 5.5: Close-ups uit figuur 5.4 met (a) de distributie vanaf de lichtbron, (b) uniforme distributie over de oppervlakte van de scene, (c) distributie vanaf de camera en (d) adaptieve distributie vanaf de camera



Photon distributie vanaf de lichtbron

Uniforme photon distributie



Photon distributie vanaf de camera

Adaptieve photon distributie vanaf de camera

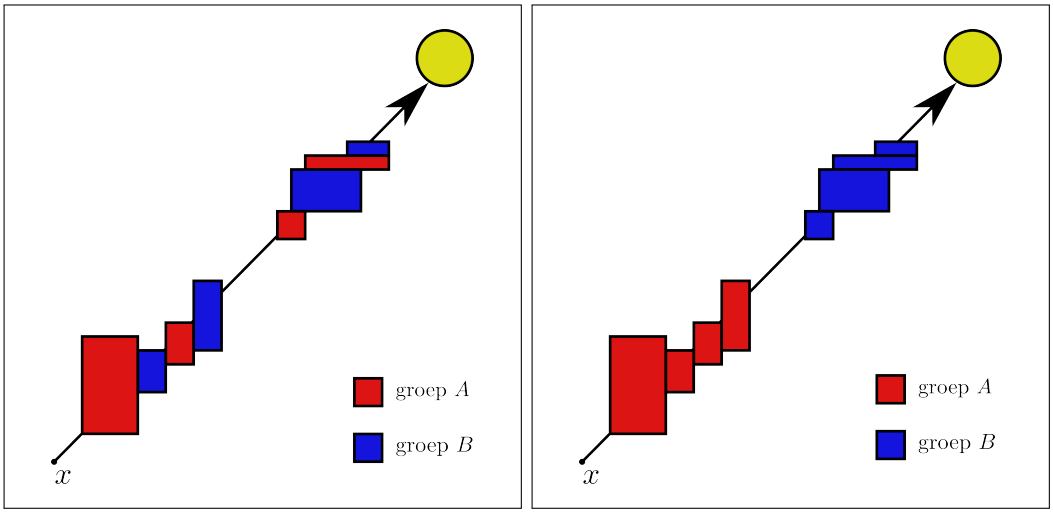
Figuur 5.6: Dichtheid van de photonen per pixel voor elke photon distributie methode.

5.3 Integratie met acceleratiestructuren

Sinds het ontstaan van ray tracing is veel onderzoek gebeurd naar het reduceren van het aantal benodigde intersectietesten. Hiervoor zijn verschillende acceleratiestructuren ontwikkeld waaronder kd-bomen, reguliere grids en Bounding Volume Hiërarchies (BVH) [7] [8].

Voorlopig beschouwden we de groepen A en B steeds als lijsten van kandidaat blokkers die we één voor één testen op intersecties. Het aantal intersectietesten kan echter gereduceerd worden door een acceleratiestructuur te gebruiken.

Om efficiëntere en compactere acceleratiestructuren te bouwen, proberen we de groepen A en B te kiezen opdat naburige kandidaat blokkers in één groep samenzitten. Om dit te bereiken sorteren we de geometrische primitieven eerst volgens hun afstand tot het punt x waar we de belichting evalueren. Deze gesorteerde lijst delen we in twee, zodat de kandidaat blokkers die het dichtste bij x liggen in groep A zitten en de kandidaat blokkers die het verste van x liggen in groep B zitten. Dit wordt geïllustreerd in figuur 5.7. Onze experimenten toonden aan dat we met behulp van de gesorteerde groepering tot 50% extra intersectietesten uitspaarden.



Willekeurige groepering Gesorteerde groepering

FIGUUR 5.7: Willekeurige en geordende groepsindeling. Wanneer we de kandidaat blokkers sorteren volgens hun afstand tot het punt x en dan in twee groepen delen, bekomen we betere acceleratiestructuren.

De volgende subsecties tonen de resultaten voor de verschillende testscenes gerenderd met de verschillende acceleratiestructuren.

5.3.1 Lijst

Scene	128 schaduwstralen		1024 schaduwstralen	
	tijd (in s)	intersectietesten	tijd (in s)	intersectietesten
Killeroos	74	2756M	317	21167M
Draak en killeroo	123	5203M	571	40847M
Icosahedronen	80	4345M	510	34358M

Tabel 5.2: Benodigde tijd en intersectietesten voor de testscenes zonder acceleratiestructuur.

5.3.2 Reguliere grids

Scene	128 schaduwstralen		1024 schaduwstralen	
	tijd (in s)	intersectietesten	tijd (in s)	intersectietesten
Killeroos	69	1479M	245	10974M
Draak en killeroo	104	1943M	363	14510M
Icosahedronen	74	2592M	440	20380M

Tabel 5.3: Benodigde tijd en intersectietesten voor de testscenes met reguliere grids.

5.3.3 Kd-bomen

Scene	128 schaduwstralen		1024 schaduwstralen	
	tijd (in s)	intersectietesten	tijd (in s)	intersectietesten
Killeroos	78	1060M	240	7603M
Draak en killeroo	127	1308M	356	9412M
Icosahedronen	68	1511M	360	11739M

Tabel 5.4: Benodigde tijd en intersectietesten voor de testscenes met kd-bomen.

5.3.4 Bounding Volume Hierarchies

Scene	128 schaduwstralen		1024 schaduwstralen	
	tijd (in s)	intersectietesten	tijd (in s)	intersectietesten
Killeroos	63	885M	213	6214M
Draak en killeroo	101	1186M	316	8438M
Icosahedronen	61	1329M	336	10291M

Tabel 5.5: Benodigde tijd en intersectietesten voor de testscenes met BVH's.

5.3.5 Bespreking

Uit de resultaten van de vorige secties zien we dat het gebruik van acceleratiestructuren de rendertijd verlaagt en het aantal intersectietesten vermindert. Daarnaast zien we dat de tijdswinst groter wordt naarmate we meer schaduwstralen doorheen de scene volgen. Dit komt omdat het opbouwen van de acceleratiestructuur tijd kost. Naarmate we meer schaduwstralen doorheen de scene volgen, wordt de tijd om de acceleratiestructuur op te bouwen verwaarloosbaar ten opzichte van de tijdswinst door het verminderde aantal intersectietesten.

Daarnaast zien we dat Bounding Volume Hierarchies de beste acceleratiestructuur is, zowel in rendertijd als in aantal intersectietesten. Kd-bomen vergen immers een grotere constructietijd [7] dan Bounding Volume Hierarchies en uit de resultaten blijkt dat de kd-bomen ook minder efficiënt zijn. Reguliere grids vergen daarentegen een zeer korte constructietijd, waardoor de rendertijden van reguliere grids dicht bij de rendertijd van BVH's liggen voor 128 schaduwstralen. Daar staat tegenover dat de reguliere grids veel slechter zijn in het reduceren van het aantal intersectietesten, waardoor BVH's in het voordeel zijn bij een groter aantal schaduwstralen.

5.4 Importance sampling

De rendering vergelijking met stochastische visibiliteit is een integraalvergelijking. Dit laat ons toe om de variantie reductie technieken uit sectie 1.4.1 te gebruiken. Hierbij focussen we ons op de importance sampling techniek.

5.4.1 Importance sampling van de visibiliteitstermen

Tot nu toe kregen de drie termen van de stochastische visibiliteitsformules 2.34, 2.40 en 2.47 een gelijke kans. We kunnen de variantie van de Monte Carlo schatter echter reduceren door andere waarden voor de kansen p_1, p_2 en p_3 te kiezen. Vanuit sectie 1.4.1 weten we dat de kansdichtheid waarmee we een monster x kiezen evenredig moet zijn aan de bijdrage van het monster tot de integraal:

$$p(x) = \frac{|f(x)|}{\int_D f(x)} \tag{5.4}$$

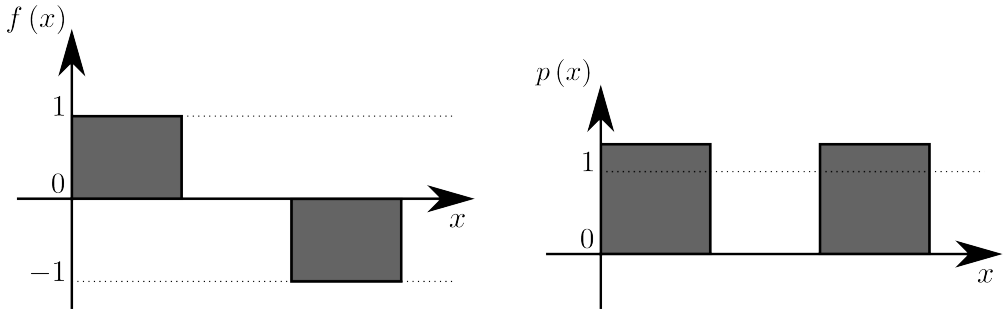
Deze optimale kansdistributiefunctie geldt echter enkel wanneer de functie $f(x)$ positief is over het gehele integratiedomein D. Wanneer dit niet het geval is, kan deze kansverdeling de convergentie vertragen.

We illustreren dat vergelijking 5.4 de convergentie vertraagt aan de hand van een voorbeeld. Stel dat we de volgende formule gebruiken om de visibiliteit te splitsen:

$$V(x,y) = V_A(x,y)\, V_B(x,y) \tag{5.5}$$

$$= V_A(x,y) + V_B(x,y) + \overline{V_A}(x,y)\, \overline{V_B}(x,y) - 1 \tag{5.6}$$

Stel daarnaast dat we de belichting van een punt x evalueren waarvoor $V_A(x,y)$ altijd gelijk is aan 1 en waarvoor $V_B(x,y)$ altijd gelijk is aan 0. De integraal



Figuur 5.8: Links de integraal die we willen bemonsteren en rechts de overeenkomstige kansdistributiefunctie.

die we willen schatten is afgebeeld in figuur 5.8 samen met zijn overeenkomstige kansdistributiefunctie.

De exacte waarde van deze integraal is gelijk aan 0. Maar de kansdistributiefunctie $p(x)$ zal nooit monsters kiezen in het gebied waar $f(x) = 0$, omdat de bijdrage van deze monsters tot de integraal gelijk is aan 0. De kansdistributiefunctie zorgt ervoor dat we enkel monsters nemen waar $f(x) = 1$ of $f(x) = -1$. Hierdoor ligt de variantie hoger dan wanneer we enkel monsters zouden nemen waar $f(x) = 0$.

Om het probleem van de negatieve waarden op te lossen, maken we de functie $f(x)$ volledig positief door er een constante term c bij op te tellen. Nu de functie $f'(x) = f(x) + c$ volledig positief is, kunnen we voor deze functie wel de optimale kansdistributiefunctie $p'(x)$ opstellen. Als I' dan de Monte Carlo schatter is voor de functie $f'(x)$ dan bekomen we de Monte Carlo schatter I voor de functie $f(x)$ door de constante $c \cdot D$ af te trekken van I'.

$$I = \sum_{i=1}^{N} \frac{f(x)}{p(x)} = \sum_{i=1}^{N} \frac{f'(x)}{p'(x)} - c \cdot D \tag{5.7}$$

Indien we deze verschuiving c toepassen op de visibiliteitstermen, dan zijn de optimale kansen voor 2.34 gelijk aan:

$$p_1 = k|c + V_A - \alpha| \qquad p_2 = k|c + V_B - \beta| \qquad p_3 = k|c + \overline{V_A} \cdot \overline{V_B} - \gamma| \tag{5.8}$$

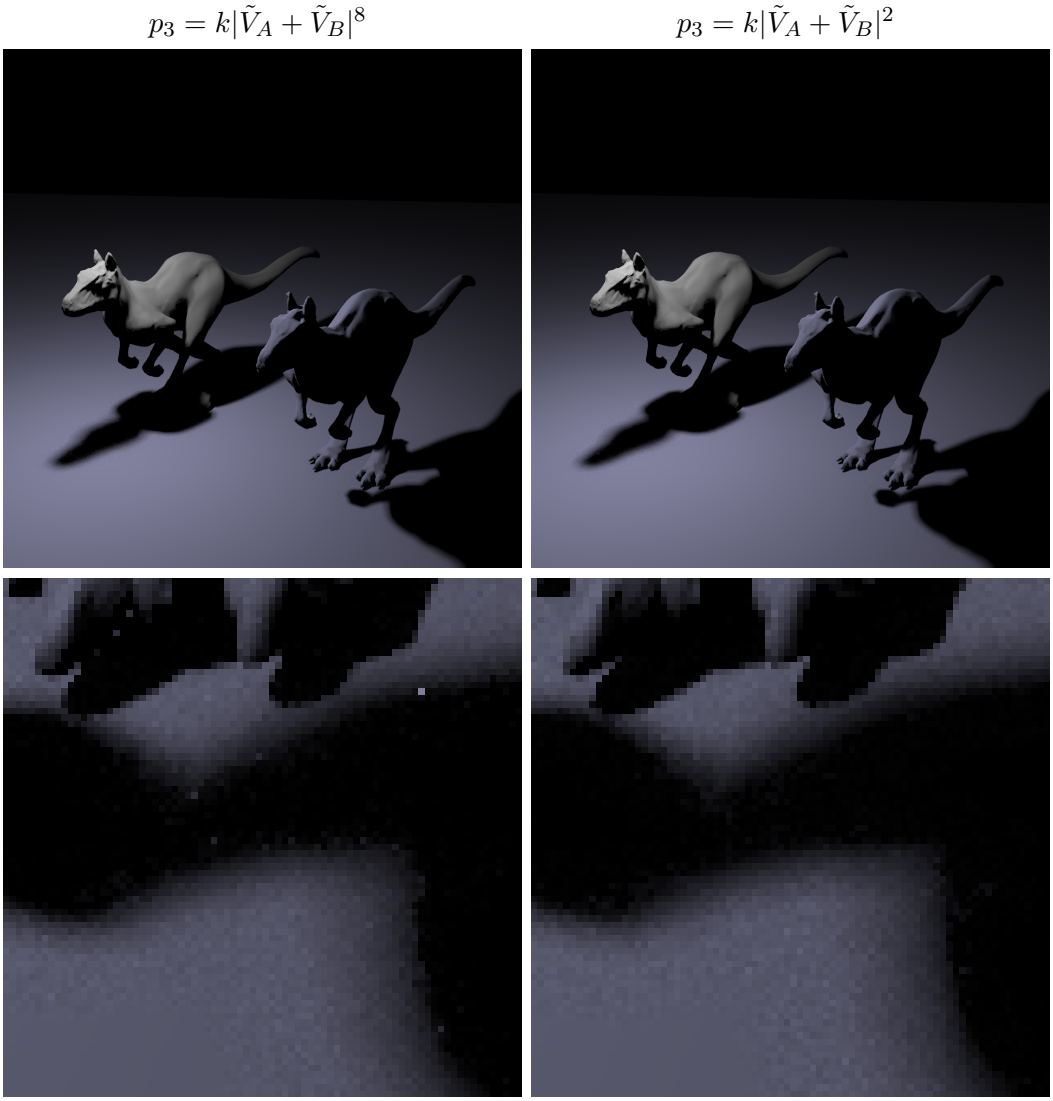
De optimale kansen voor formule 2.40 zijn:

$$p_1 = k|c + V_A| \qquad p_2 = k|c + V_B| \qquad p_3 = k|c + (V_A - V_B)^2| \tag{5.9}$$

De optimale kansen voor formule 2.47 zijn:

$$p_1 = k|c + V_A| \qquad p_2 = k|c + V_B| \qquad p_3 = k|c + (V_A + V_B)^2| \tag{5.10}$$

Merk op dat we de term $(V_A + V_B)$ in formule 5.10 slechts verheffen tot de 2de macht en niet tot de 8ste macht. Uit onze experimenten bleek dat twee eerste termen een te kleine kans krijgen wanneer we $(V_A + V_B)$ in 5.10 tot de 8ste macht verheffen. Wanneer de eerste term of de tweede term dan toch gekozen wordt, krijgen deze een enorme bijdrage omdat p_1 en p_2 zeer klein zijn (zie figuur 5.9).



FIGUUR 5.9: Lichtvlekken onstaan door de kleine kansen p_1 en p_2. Wanneer de kans p_3 zeer groot wordt, worden de kansen p_1 en p_2 zeer klein. Bij het kiezen van een term met kans p_1 of p_2 moeten we de bijdrage van die term delen door de kans. Hierdoor wordt de bijdrage van de term enorm, waardoor er lichtvlekken ontstaan in de resulterende afbeelding.

Om een schatting te bekomen voor V_A en V_B passen we een adaptief schema toe. Voor de eerste 128 schaduwstralen kiezen we de kansen $p_1 = p_2 = p_3 = 1/3$. Bij deze eerste 128 schaduwstralen onthouden we hoe vaak we groep A en B kozen om te intersecteren en hoe vaak we een primitief uit groep A en groep B snijden. Met behulp van deze gegevens kunnen we de waardes van V_A en V_B als volgt schatten:

$$\tilde{V}_A = 1 - \frac{A_{geraakt}}{A_{gekozen}} \qquad \tilde{V}_B = 1 - \frac{B_{geraakt}}{B_{gekozen}} \tag{5.11}$$

5.4.2 Resultaten

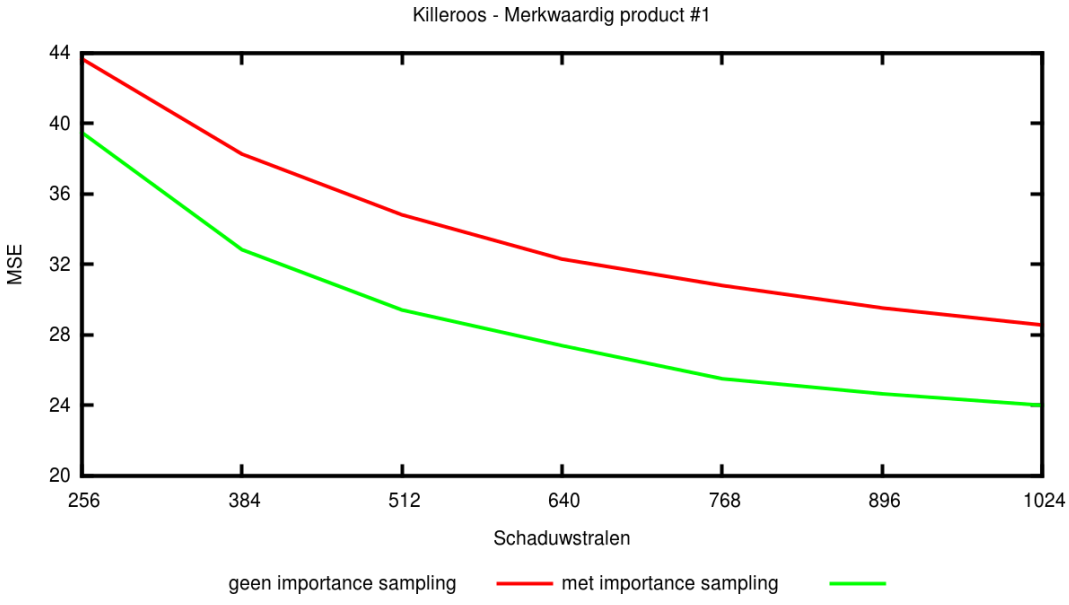
Deze sectie geeft de resultaten van het bovenstaande importance sampling schema in de verschillende testscenes. Hierbij renderden we de drie testscenes met de drie verschillende visibiliteitsdecomposities met respectievelijk 256, 384, 512, 640, 768, 896 en 1024 schaduwstralen en 4 monsters per pixel. Bovendien werd iedere afbeelding gerenderd met en zonder het importance sampling schema. Om de kwaliteit van de afbeeldingen met en zonder importance sampling te vergelijken, berekenden we voor de twee afbeeldingen de Mean Squared Error ten opzichte van een referentie afbeelding. Hierbij definieerden we de MSE als volgt:

$$MSE = \frac{1}{n_x n_y} \sum_{x=1}^{n_x} \sum_{y=1}^{n_y} (\hat{r}_{xy} - r_{xy})^2 + (\hat{g}_{xy} - g_{xy})^2 + \left(\hat{b}_{xy} - b_{xy}\right)^2 \tag{5.12}$$

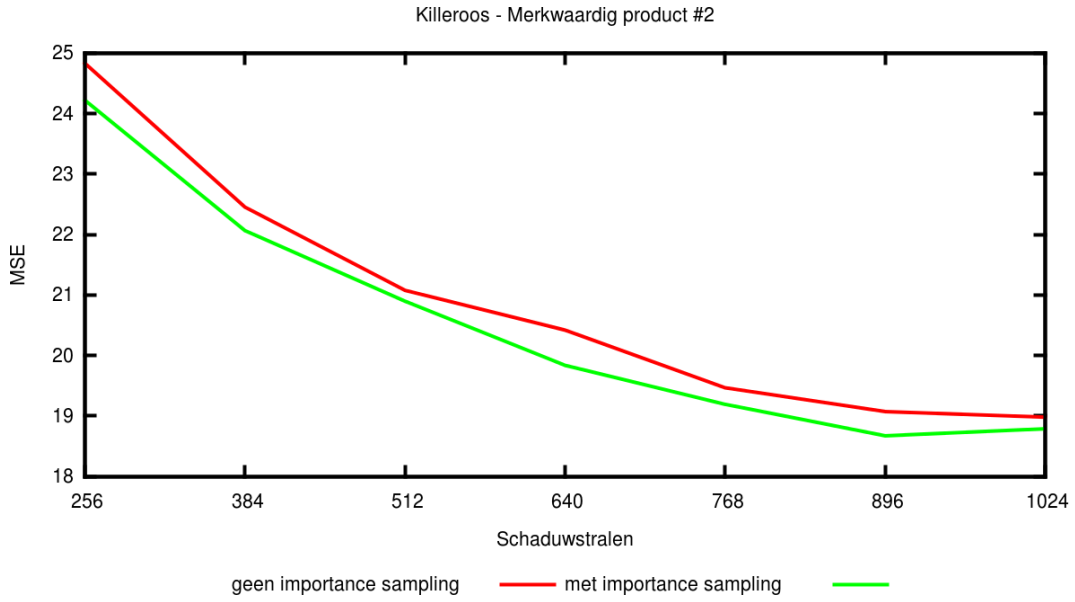
r_{xy}, g_{xy} en b_{xy} zijn respectievelijk de rode, groene en blauwe kleurcomponent van de pixel op (x, y) van de gerenderde afbeelding. $\hat{r}_{xy}$, $\hat{g}_{xy}$ en $\hat{b}_{xy}$ zijn respectievelijk de rode, groene en blauwe kleurcomponent van de pixel op (x, y) van de referentie afbeelding.

De referentie afbeeldingen zijn gerenderd met behulp van een standaard path tracing algoritme met 16 schaduwstralen per pixel en 4096 schaduwstralen.

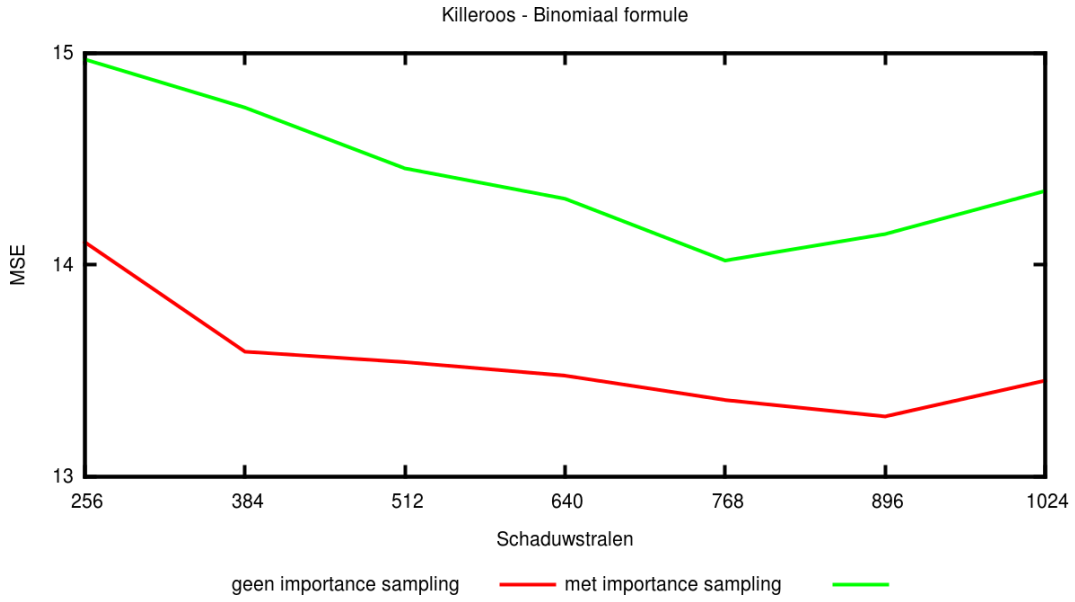
Killeroos scene



Figuur 5.10: Killeroos scene: invloed importance sampling met merkwaardig product #1

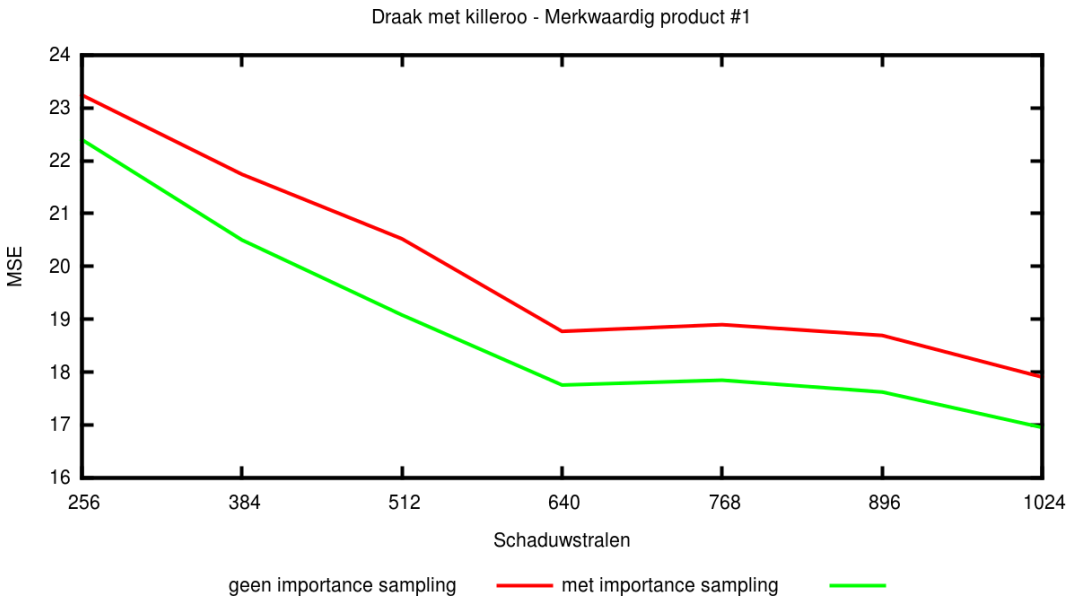


FIGUUR 5.11: Killeroos scene: invloed importance sampling met merkwaardig product #2

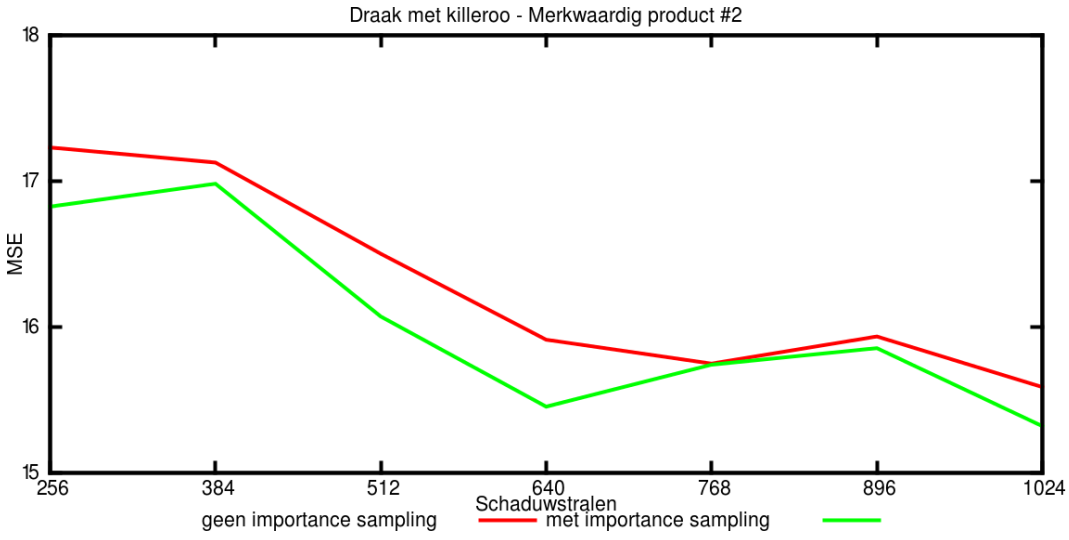


FIGUUR 5.12: Killeroos scene: invloed importance sampling met de binomiaal formule

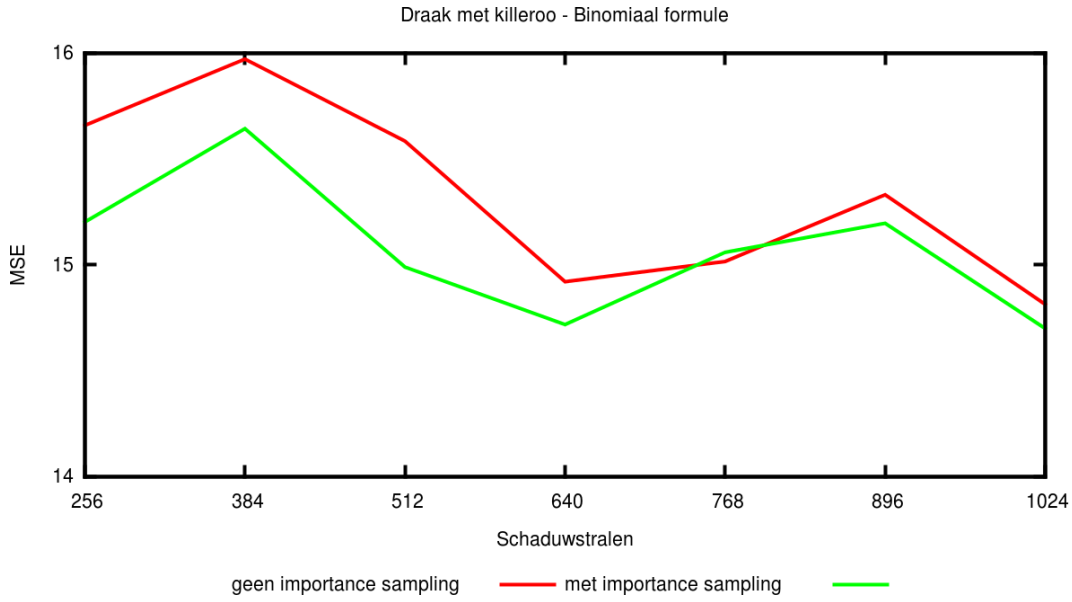
Draak met Killeroo scene



Figuur 5.13: Draak met Killeroo scene: invloed importance sampling met merkwaardig product #1

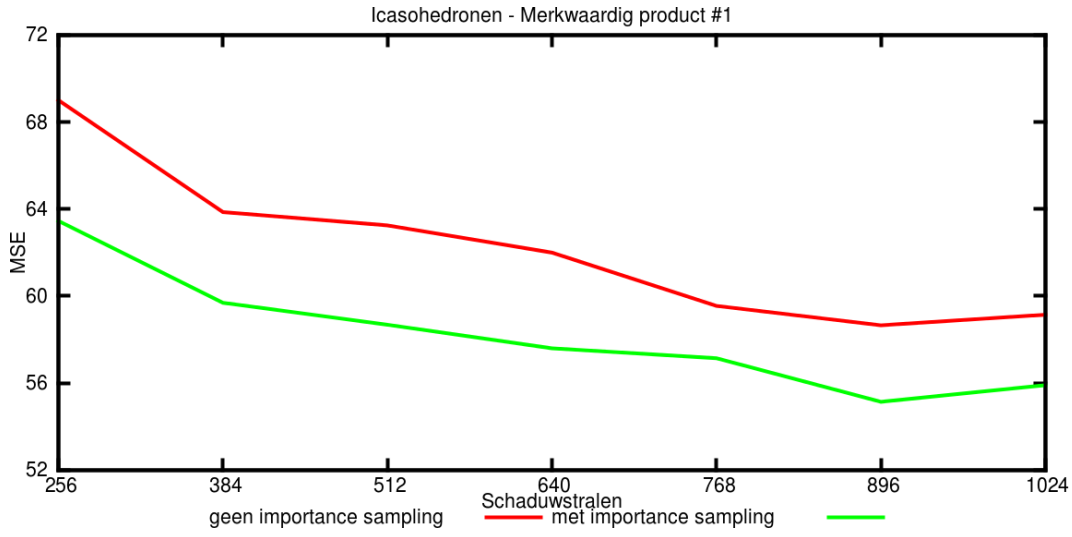


Figuur 5.14: Draak met Killeroo scene: invloed importance sampling met merkwaardig product #2

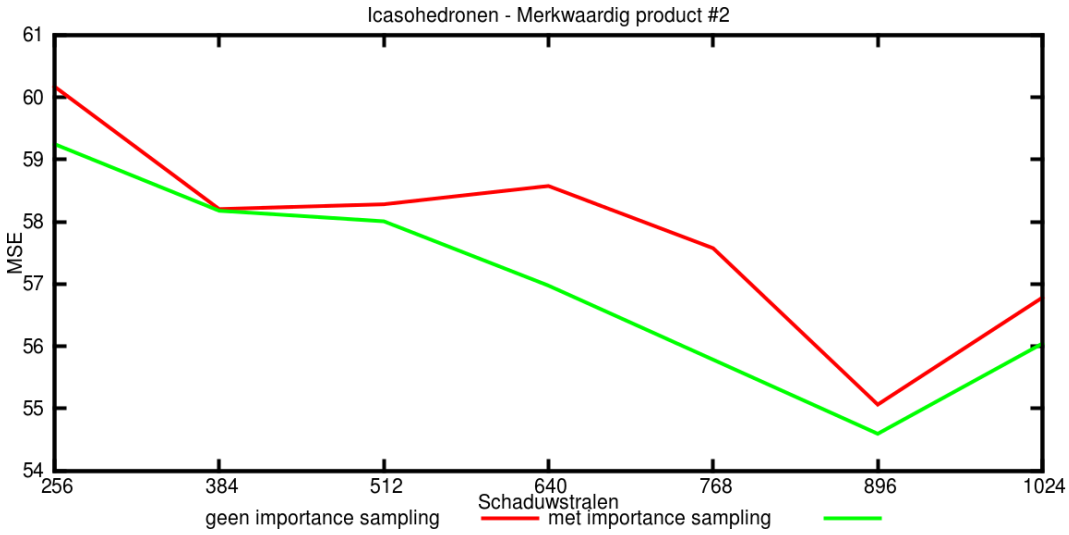


FIGUUR 5.15: Draak met Killeroo scene: invloed importance sampling met de binomiaal formule

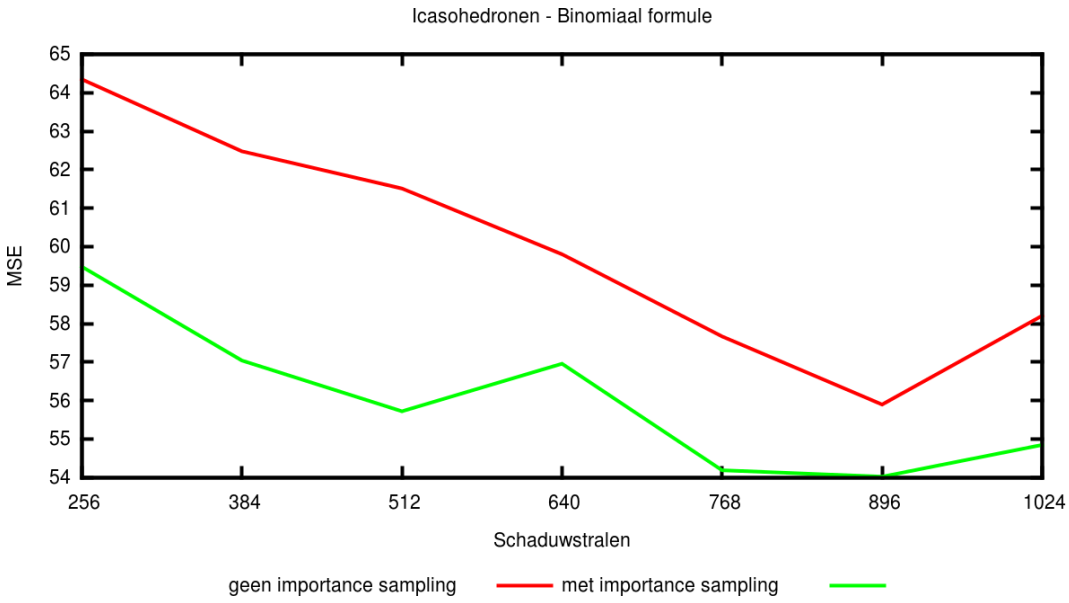
Cornell box met icosahedronen



FIGUUR 5.16: Cornell box met icosahedronen: invloed importance sampling met merkwaardig product #1



Figuur 5.17: Cornell box met icosahedronen: invloed importance sampling met merkwaardig product #2



Figuur 5.18: Cornell box met icosahedronen: invloed importance sampling met de binomiaal formule

5.4.3 Bespreking

De figuren uit sectie 5.4.2 tonen de MSE ten opzichte van het aantal schaduwstralen voor de drie testscenes. Hierbij maken we de vergelijking tussen afbeeldingen gerendered met en zonder importance sampling. Uit de grafieken zien we dat het importance sampling schema over het algemeen een gunstige invloed heeft op de convergentie van de afbeeldingen.

Het importance sampling schema heeft enkel een negatief effect op de convergentie wanneer we de Killeroos scene renderen met behulp van de binomiaal formule (zie figuur 5.12).

In sommige grafieken zien we ook de MSE stijgen naarmate dat we meer schaduwstralen schieten. Dit is niet geheel onverwacht vermits de occlusion map met volumetrische occluders er toch niet geheel in slaagt om al de occluders op te nemen. Hierdoor zal het stochastisch algoritme niet kunnen convergeren naar de hoge kwaliteit referentie afbeelding en kan de MSE stijgen.

5.5 Analytische Integratie

Met behulp van de occlusion map kunnen we onderscheiden wanneer een punt volledig belicht is, volledig in schaduw ligt of in een zacht schaduwgebied ligt. Wanneer we enkel licht photonen vinden rond een punt x, waarvan we de belichting willen berekenen, dan gaan we ervan uit dat de visibiliteit $V(x, y)$ altijd gelijk is aan 1. Hierdoor reduceert de rendering vergelijking van de directe belichting tot:

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{A_{licht}} f_r(x, \overrightarrow{xy} \leftrightarrow \Theta)\, L(y \to \overrightarrow{yx})\, G(x, y)\, dA_y \quad (5.13)$$

Deze integraal kan voor eenvoudige belichtingsituaties ook analytisch berekend worden. Indien de scene enkel bestaat uit diffuse polygonale oppervlakken en lichtbronnen, dan kunnen we vergelijking 5.13 schrijven als:

$$L(x) = L_e(x) + \int_{A_{licht}} f_r(x)\, L(y)\, G(x, y)\, dA_y \quad (5.14)$$

Voor deze integraal bestaat de volgende gesloten formule [1]:

$$L(x) = \frac{M}{2\pi} \sum_{i=1}^{n} \Theta_i(x)\, (\Gamma_i(x) \cdot N) \quad (5.15)$$

Vergelijking 5.15 berekent de belichting van een polygonale lichtbron die M $watt/m^2$ uitstraalt, bestaande uit n hoekpunten $\{v_1, v_2, ..., v_n\}$. Θ_i is hierbij gelijk aan de booglengte van de lijn $v_i - v_{i+1}$ geprojecteerd op de hemisfeer rond het punt x. Γ_i is gelijk aan het kruisproduct $\frac{(v_i - x) \times (v_{i+1} - x)}{||(v_i - x) \times (v_{i+1} - x)||}$. N is de normaal in het punt x.

Vergelijking 5.15 is zeer efficiënt om te evalueren in tegenstelling tot het bemonsteren van de lichtbron. Bovendien berekent 5.15 de exacte oplossing van de belichting.

5.5.1 Resultaten

Deze sectie toont de invloed van het analytische integratieschema op de rendertijd. We renderden de drie testscenes met stochastische visibiliteit waarbij we gebruik maakten van merkwaardig product #1 om de visibiliteit te ontbinden. In de volledig belichte gebieden van de scene evalueren we de belichting met analytische en Monte Carlo integratie. Tabel 5.6 toont de invloed van het analytische integratie schema op de rendertijd.

Scene	**256 schaduwstralen**		**1024 schaduwstralen**	
	Analytisch	Monte Carlo	Analytisch	Monte Carlo
Killeroos	85s	54s (-36%)	210s	88s (-58%)
Draak met killeroo	128s	80s (-37%)	312s	108s (-65%)
Icosahedronen	95s	81s (-15%)	315s	255s (-19%)

Tabel 5.6: Invloed van analytische integratie op de render tijd.

5.5.2 Bespreking

Uit tabel 5.6 zien we duidelijk dat het analytische integratie schema een positieve invloed heeft op de rendertijd. De reden hiervoor is dat vergelijking 5.15 veel eenvoudiger is om te evalueren dan het bemonsteren van de lichtbron met een groot aantal monsters.

Het effect van de analytische integratie wordt ook sterker naarmate we meer schaduwstralen gebruiken voor de Monte Carlo schatter. De kost om de integraal analytisch te berekenen blijft constant, terwijl de kost om de integraal met Monte Carlo te schatten linear stijgt met het aantal schaduwstralen.

Daarnaast zien we dat het effect van de analytische integratie ook sterker is wanneer de scene grote volledig belichte gebieden bevat. De Cornell box met icosahedronen wordt vooral gedomineerd door zachte schaduwen van de vele icosahedronen. Hierdoor is het effect van de analytische integratie het kleinst voor deze testscene.

Een groot nadeel van het analytische integratie schema is dat zowel de oppervlakken als de lichtbronnen diffuus moeten zijn. Voor vele brdf's is het onmogelijk om de integraal 5.13 analytisch uit te rekenen.

5.6 Besluit

In dit hoofdstuk hebben we een groot aantal uitbreidingen voorgesteld voor het naïeve algoritme uit hoofdstuk 4. In dit hoofdstuk zagen we dat de occlusion map er vaak niet in slaagt al de kandidaat blokkers op te slaan. De kans is immers groot dat de fijne geometrie in de scene niet opgenomen wordt in de occlusion map. In dit hoofdstuk hebben we deze problematiek op twee manieren aangepakt:

- **Volumetrische occluders**: we introduceren extra geometrie in de scene die compenseert voor het missen van de fijne geometrie.
- **Alternatieve photon distributie**: we distribueren meer photonen in de gebieden waar we stochastische visibiliteit gaan toepassen.

Daarnaast reduceerden we het benodigde aantal intersectietesten door acceleratiestructuren te bouwen over de groepen A en B. We constateerden hierbij dat de tijdwinst behaald door het verminderd aantal intersectietesten ruimschoots compenseerde voor de tijd benodigd voor het opbouwen van de acceleratie structuren. Bovendien wordt de constructietijd van de acceleratiestructuren verwaarloosbaar wanneer we voldoende schaduwstralen schieten. Van de geteste acceleratiestructuren zagen we dat de Bounding Volume Hierarchie over het algemeen het beste presteerde.

Voor de volledig belichte gebieden stelden we voor om de belichting analytisch te integreren. Hierdoor vermijden we dat we de lichtbron moeten bemonsteren met een groot aantal schaduwstralen, wat een grote reductie in rendertijd betekent. Daarnaast toonden we aan dat de reductie in rendertijd groter wordt wanneer we meer schaduwstralen door de scene volgen en wanneer de scene veel volledig belichte gebieden bevat.

Finaal pasten we ook importance sampling toe op de verschillende visibiliteitstermen waardoor we de convergentie van de Monte Carlo schatter verbeterden.

Hoofdstuk 6

Het uitgebreide stochastische algoritme

In dit hoofdstuk introduceren we het uitgebreide stochastische algoritme. Hierbij combineren we het naëive algoritme uit hoofdstuk 4 met de optimale uitbreidingen uit hoofdstuk 5. In de eerste sectie bespreken we welke uitbreidingen we integreren met het naïeve algoritme. Vervolgens evalueren we de performantie door het uitgebreide algoritme te vergelijken met het occlusion map algoritme en een path tracing algoritme.

6.1 Beschrijving van het uitgebreide stochastische algoritme

In deze sectie motiveren we welke uitbreidingen uit hoofdstuk 5 we toevoegen aan het naïeve algoritme uit hoofdstuk 4.

Voor we beginnen met de evaluatie van de belichting, construeren we de volumetrische occluders voor elk waterdicht model in de scene. Door de de waterdichte modellen op te vullen met volumetrische occluders, introduceren we meer kandidaat blokkers die compenseren voor de gemiste fijne geometrie in de scene.

Daarna creëren we de occlusion map. Hierbij gebruiken we het adaptieve camera distributie schema uit sectie 5.2.3. Hoewel we de occlusion map opnieuw moeten opbouwen wanneer we dezelfde scene willen renderen vanuit een ander camera standpunt, verdeelt deze distributie methode de photonen optimaal voor ons algoritme. Door de adaptieve distributie bevinden er zich meer occlusion photonen in de zachte schaduwgebieden van de scene, waardoor we een betere schatting krijgen van de kandidaat blokkers in deze gebieden.

Vervolgens renderen we de afbeelding door zichtstralen doorheen de scene te volgen. Wanneer we de belichting evalueren voor een punt x zoeken we de $n_{occlusion}$ dichtstbijzijnde occlusion photonen en de n_{licht} dichtstbijzijnde licht photonen. Afhankelijk van het aantal occlusion photonen en licht photonen gaat de belichtingsevaluatie verder als volgt:

- **enkel occlusion photonen**: we gaan er van uit dat we ons in een schaduwgebied bevinden. De belichting in het punt x is bijgevolg gelijk aan nul.

- **enkel licht photonen**: we gaan er van uit dat we ons in een volledig belicht gebied bevinden. We evalueren de belichting met behulp van analytische integratie in diffuse scenes of met Monte Carlo integratie in complexere scenes.

- **occlusion photonen en licht photonen**: we evalueren de belichting met behulp van stochastische visibiliteit.

Voor de stochastische visibiliteitsevaluatie gebruiken we de kandidaat blokkers in de gevonden occlusion photonen. Deze verzameling $Z = \{z_1, z_2, ..., z_n\}$ van kandidaat blokkers delen we in twee even grote groepen A en B. Hiervoor sorteren we de kandidaat blokkers op basis van hun afstand tot het punt x waarbij groep A de kandidaat blokkers bevat die het dichtste bij x liggen en groep B de kandidaat blokkers die het verste van x liggen. Vervolgens bouwen we een BVH over de groepen A en B. In sectie 5.3 zagen we immers dat de Bounding Volume Hierarchie de grootste reductie in rendertijd en aantal intersectietesten gaf.

Verder gebruiken we het importance sampling schema uit 5.4 om de variantie van de Monte Carlo integratie te verminderen.

6.2 Resultaten

In deze sectie zullen we het volledige algoritme evalueren door het toe te passen om zes testscenes met variërende complexiteit in de visibiliteit. Daarnaast zullen we het stochastische visibiliteitsalgoritme vergelijken met twee andere belichtingsalgoritmen.

6.2.1 Setup

De afbeeldingen van de scene worden gerenderd met vier monsters per pixel en met respectievelijk 256, 512 en 1024 schaduwstralen. Om de parameters van het volledige algoritme te bepalen steunen we op de bevindingen uit sectie 4.2. De parameters voor de zes testscenes zijn terug te vinden in tabel 6.1.

Scene	$\mathbf{A_{scene}}$	$\mathbf{n_{photonen}}$	$\mathbf{n_{zoek}}$	r
Killeroo's	1033580	3000000	100	3.3
Draak met Killeroo	1864116	3000000	100	4.5
Cornell met icosahedronen	1755910	3000000	100	4.31
Menger vlak	6463	3000000	100	0.262
Yeah right	6839	3000000	100	0.270
Sponza	7518	3000000	100	0.282

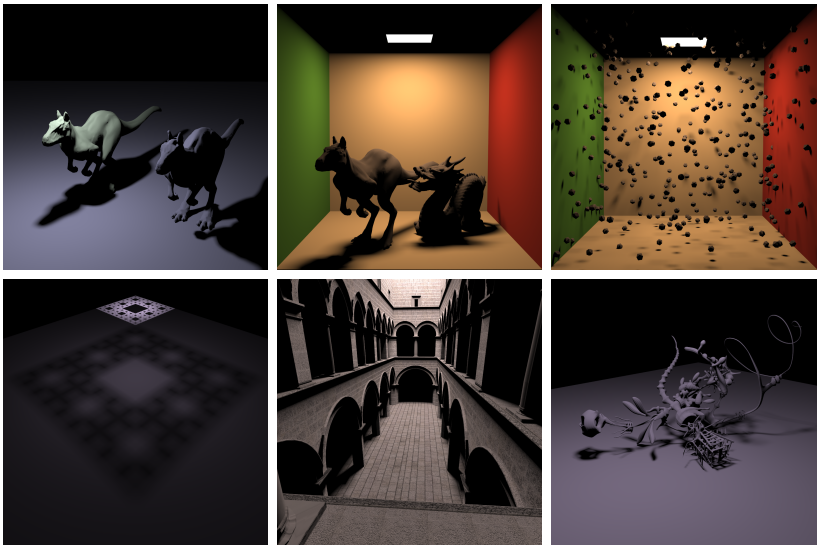
Tabel 6.1: Parameters voor de verschillende testscenes

Om de performantie van het complete algoritme uit sectie 6.1 te schatten, vergelijken we het uitgebreide stochastische algoritme met het naïeve algoritme en met twee deterministische visibiliteitsalgoritmen:

- **het occlusion map algoritme** is volledig analoog aan het uitgebreide stochastische algoritme, maar zal geen stochastische visibiliteit toepassen. De visibiliteit zal enkel geëvalueerd worden wanneer er een mix van occlusion en licht photonen gevonden wordt rond het punt dat we willen belichten. Het occlusion map bouwt vervolgens een Bounding Volume Hierarchie over de volledige verzameling van kandidaat blokkers uit al de gevonden occlusion photonen. Deze BVH zal vervolgens voor elke schaduwstraal gebruikt worden om de visibiliteit te evalueren.

- **het path tracing algoritme met de occlusion map** zal de occlusion map gebruiken om te testen of het punt dat we willen evalueren compleet belicht is, compleet in de schaduw ligt of in een zacht schaduwgebied ligt. Wanneer het punt volledig belicht is, gebruiken we analytische integratie om de belichting te berekenen. Wanneer we in de zachte schaduwgebieden zitten zal het path tracing algoritme een BVH gebruiken om de visibiliteit te evalueren. Deze BVH is alvorens het renderen opgebouwd over de gehele geometrie van de scene.

De zes testscenes zijn afgebeeld in figuur 6.1. Deze zijn respectievelijk de Killeroos scene, Draak met Killeroo, Cornell box met icosahedronen, Menger vlak scene, Sponza scene en het Yeah right model. De Menger vlak scene bevat een fractaal oppervlak met zeer nauwe mazen die voor een complexe visibiliteitssituatie zorgen. De Sponza scene toont hoe de algoritmes zich gedragen in een grote scene. Finaal zorgt het zeer onregelmatige Yeah right model ook voor zeer complexe visibiliteitsevents.



FIGUUR 6.1: Afbeelding van de zes testscenes

6.2.2 Killeroos

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	50	738	12.11
Occlusion map	-	52	594	14.36
Stochastische visibiliteit	product #1	54	574	40.28
	product #2	53	590	25.57
	binomiaal formule	54	640	16.25
Naïef algoritme	binomiaal formule	74	4269	55.65

Tabel 6.2: Rendertijd en aantal intersectie testen voor de Killeroos scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	66	1349	11.89
Occlusion map	-	62	1065	14.2
Stochastische visibiliteit	product #1	66	1023	30.51
	product #2	65	1050	21.99
	binomiaal formule	70	1205	15.74
Naïef algoritme	binomiaal formule	133	8513	54.93

Tabel 6.3: Rendertijd en aantal intersectie testen voor de Killeroos scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	99	2572	11.82
Occlusion map	-	85	2007	14.05
Stochastische visibiliteit	product #1	87	1921	24.73
	product #2	89	1975	19.59
	binomiaal formule	98	2336	15.18
Naïef algoritme	binomiaal formule	258	17002	54.76

Tabel 6.4: Rendertijd en aantal intersectie testen voor de Killeroos scene met 1024 schaduwstralen

6.2.3 Draak met Killeroo

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	63	619	7.57
Occlusion map	-	76	494	14.56
Stochastische visibiliteit	product #1	79	474	22.44
	product #2	79	484	17.19
	binomiaal formule	78	512	15.5
Naïef algoritme	binomiaal formule	87	3465	215.23

Tabel 6.5: Rendertijd en aantal intersectie testen voor de Draak met Killeroo scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	77	1091	7.47
Occlusion map	-	85	838	14.81
Stochastische visibiliteit	product #1	88	801	18.96
	product #2	89	813	16.14
	binomiaal formule	93	902	14.97
Naïef algoritme	binomiaal formule	159	6908	216.34

Tabel 6.6: Rendertijd en aantal intersectie testen voor de Draak met Killeroo scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	108	2025	7.51
Occlusion map	-	104	1532	14.61
Stochastische visibiliteit	product #1	109	1449	17.02
	product #2	111	1476	15.54
	binomiaal formule	118	1670	14.8
Naïef algoritme	binomiaal formule	293	13773	214.73

Tabel 6.7: Rendertijd en aantal intersectie testen voor de Draak met Killeroo scene met 1024 schaduwstralen

6.2.4 Cornell box met icosahedronen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	115	2401	30.96
Occlusion map	-	78	1941	57.93
Stochastische visibiliteit	product #1	81	1869	67.06
	product #2	82	1915	60.75
	binomiaal formule	88	1964	63.15
Naïef algoritme	binomiaal formule	120	6536	73.45

Tabel 6.8: Rendertijd en aantal intersectie testen voor de icosahedronen scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	216	4748	33.84
Occlusion map	-	135	3833	55.4
Stochastische visibiliteit	product #1	140	3697	59.97
	product #2	144	3770	58.08
	binomiaal formule	157	3931	56.99
Naïef algoritme	binomiaal formule	224	13043	67.78

Tabel 6.9: Rendertijd en aantal intersectie testen voor de icosahedronen scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	417	9449	29.89
Occlusion map	-	264	7633	55.21
Stochastische visibiliteit	product #1	260	7349	57.4
	product #2	265	7482	55.18
	binomiaal formule	299	7865	58.42
Naïef algoritme	binomiaal formule	434	26064	64.91

Tabel 6.10: Rendertijd en aantal intersectie testen voor de icosahedronen scene met 1024 schaduwstralen

6.2.5 Menger vlak

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	60	1087	6.81
Occlusion map	-	71	1053	13.1
Stochastische visibiliteit	product #1	82	951	14.49
	product #2	84	990	13.81
	binomiaal formule	86	1037	13.57
Naïef algoritme	binomiaal formule	158	6553	26.41

Tabel 6.11: Rendertijd en aantal intersectie testen voor de Menger vlak scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	110	2147	6.73
Occlusion map	-	123	2082	12.97
Stochastische visibiliteit	product #1	155	1875	13.52
	product #2	157	1947	13.31
	binomiaal formule	163	2089	13.2
Naïef algoritme	binomiaal formule	307	13086	26.11

Tabel 6.12: Rendertijd en aantal intersectie testen voor de Menger vlak scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	205	4265	6.68
Occlusion map	-	224	4131	13.05
Stochastische visibiliteit	product #1	311	3718	13.49
	product #2	300	3864	13.24
	binomiaal formule	314	4198	13.15
Naïef algoritme	binomiaal formule	605	26165	25.83

Tabel 6.13: Rendertijd en aantal intersectie testen voor de Menger vlak scene met 1024 schaduwstralen

6.2.6 Sponza

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	96	1832	376.2
Occlusion map	-	78	1636	377.67
Stochastische visibiliteit	product #1	81	1584	453.9
	product #2	83	1634	433.64
	binomiaal formule	86	1710	385.02
Naïef algoritme	binomiaal formule	79	3181	694.67

Tabel 6.14: Rendertijd en aantal intersectie testen voor de Sponza scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	163	3578	375.7
Occlusion map	-	119	3193	377.09
Stochastische visibiliteit	product #1	124	3088	427.52
	product #2	126	3196	417.04
	binomiaal formule	135	3423	382.77
Naïef algoritme	binomiaal formule	137	6340	686.85

Tabel 6.15: Rendertijd en aantal intersectie testen voor de Sponza scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	299	7092	376.2
Occlusion map	-	200	6326	376.8
Stochastische visibiliteit	product #1	207	6105	409.59
	product #2	213	6316	405.32
	binomiaal formule	232	6859	380.41
Naïef algoritme	binomiaal formule	257	12619	687.85

Tabel 6.16: Rendertijd en aantal intersectie testen voor de Sponza scene met 1024 schaduwstralen

6.2.7 Yeah right

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	49	630	7.7
Occlusion map	-	57	508	13.99
Stochastische visibiliteit	product #1	60	489	23.11
	product #2	60	500	17.58
	binomiaal formule	61	531	14.76
Naïef algoritme	binomiaal formule	83	4556	78.89

Tabel 6.17: Rendertijd en aantal intersectie testen voor de Yeah right scene met 256 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	68	1172	7.78
Occlusion map	-	68	926	13.82
Stochastische visibiliteit	product #1	72	890	19.35
	product #2	72	906	16.28
	binomiaal formule	77	1003	14.25
Naïef algoritme	binomiaal formule	149	9085	78.58

Tabel 6.18: Rendertijd en aantal intersectie testen voor de Yeah right scene met 512 schaduwstralen

algoritme	**formule**	**tijd** (in seconden)	**intersectietesten** (x1000000)	**mse**
Path tracing	-	102	2251	7.59
Occlusion map	-	90	1762	13.53
Stochastische visibiliteit	product #1	95	1686	17.35
	product #2	97	1719	15.29
	binomiaal formule	107	1945	14.03
Naïef algoritme	binomiaal formule	280	18178	78.26

Tabel 6.19: Rendertijd en aantal intersectie testen voor de Yeah right scene met 1024 schaduwstralen

6.3 Bespreking

De tabellen uit de vorige sectie tonen de rendertijd, het aantal intersectietesten en de MSE van de afbeeldingen gerenderd met de verschillende algoritmes. De MSE is steeds berekend ten opzichte van een hoge kwaliteit referentie afbeelding gerenderd met 4096 schaduwstralen en 16 monsters per pixel.

6.3.1 MSE

Uit de resultaten zien we dat de afbeeldingen gerenderd met het path tracing algoritme steeds de kleinste MSE hebben. Dit is logisch aangezien het path tracing algoritme enkel gebruik maakt van de occlusion map om de verschillende gebieden in de scene te identificeren. Om de visibiliteit te evalueren gebruikt het path tracing algoritme een BVH gebouwd over de geometrie van de hele scene. Hierdoor zal het path tracing algoritme nooit kandidaat blokkers missen, wat wel het geval is bij de andere algoritmes. De andere algoritmes gebruiken de occlusion map om de kandidaat blokkers van een punt x te zoeken. Zoals reeds besproken is de kans groot dat een deel van de fijne geometrie in de scene gemist wordt, waardoor deze geometrie nooit visibiliteitsevents zal genereren.

Daarnaast zien we dat de MSE van de afbeeldingen gerenderd met het uitgebreide stochastische algoritme steeds groter is dan de MSE van de afbeeldingen gerenderd met het occlusion map algoritme. Dit is compleet volgens de verwachting vermits het occlusion map algoritme de visibiliteit deterministisch evalueert, terwijl het uitgebreide stochastische algoritme de visibiliteit stochastisch evalueert. De MSE behaald met het occlusion map algoritme is dus de ondergrens die we kunnen behalen met het uitgebreide stochastische algoritme.

Zoals verwacht zien we dat de MSE van het uitgebreide stochastische algoritme het kleinste is wanneer we de binomiaal formule gebruiken. Daarnaast zien we dat de MSE bekomen met de ontbinding volgens merkwaardig product #2 steeds kleiner is dan de MSE bekomen met de ontbinding volgens merkwaardig product #1. Dit bevestigt de theoretische afleidingen van de variantie uit hoofdstuk 2.

Finaal merken we op dat de MSE van het naïeve algoritme het grootst is. Dit komt doordat het naïeve algoritme geen gebruik maakt van de volumetrische occluders of van de adaptieve photon distributie methode. Het naïeve algoritme mist hierdoor een groot deel van de fijne geometrie in de scene waardoor deze nooit kan convergeren naar het juiste resultaat.

6.3.2 Intersectietesten

Het naïeve algoritme vereist steeds het grootste aantal intersectietesten. Dit is ook het enige algoritme dat geen gebruik maakt van acceleratie structuren om de visibiliteitsevaluatie te versnellen.

Na het naïeve algoritme vereist het path tracing algoritme het hoogste aantal intersectietesten. Dit kunnen we verklaren doordat de BVH die gebruikt wordt om de zichtbaarheid te bepalen, gebouwd is voor de gehele geometrie van de scene. Deze

BVH zal over het algemeen veel groter zijn, waardoor we vaak diep in de boom moeten afdalen en meer geometrie moeten testen op intersecties.

Wanneer we het aantal intersectietesten van het occlusion map algoritme vergelijken met het aantal intersectietesten van het uitgebreide stochastische algoritme, dan valt op dat deze zeer dicht bij elkaar liggen. Dit is een groot verschil met de resultaten uit sectie 4.4 waar het naïeve algoritme gemiddeld 26% minder intersectietesten nodig had dan het occlusion map algoritme. In het naïeve algoritme bestaan de groepen A en B uit lijsten van kandidaat blokkers. Deze lijsten moeten steeds doorlopen worden totdat we een eerste intersectie vinden met een kandidaat blokker. Vermits de kandidaat blokkers vaak gemist worden, duurt het dikwijls lang om in de lijst een eerste intersectie te vinden. Het naïeve algoritme bespaart gemiddeld 26% van de intersectietesten omdat we dankzij de stochastische visibiliteit vaak maar één groep moeten testen, terwijl het occlusion map algoritme steeds groep A en groep B moet testen.

Het bovenstaande voordeel wordt echter teniet gedaan door de introductie van de BVH. Deze acceleratiestructuur is zeer efficiënt in het vinden van een eerste intersectie waardoor het testen van één of twee groepen ongeveer evenveel intersectietesten vereist.

Daarnaast zien we dat het benodigde aantal intersectietesten steeds het laagst is wanneer we het merkwaardig product #1 gebruiken en het hoogst wanneer we de binomiaal formule gebruiken. Dit kunnen we verklaren als volgt:

- Bij het merkwaardig product #1 kunnen we de evaluatie van het visibiliteitsproduct $\overline{V}_A \cdot \overline{V}_B$ immers afbreken wanneer $V_A = 1$ of wanneer $V_B = 1$. Het product is dan automatisch gelijk aan 0.

- Bij het merkwaardig product #2 zal de bijdrage van de term waarbij we de twee groepen moeten testen, $(V_A - V_B)^2$, slechts een kleine bijdrage hebben wanneer $V_A \simeq V_B$. Daarom geeft het importance schema deze term een kleine kans en de termen die slechts één groep testen een grote kans.

- Bij de binomiaal formule is de bijdrage van de term $(V_A + V_B)^8$ zeer groot. Hierdoor zal het importance sampling schema een grote kans toekennen aan deze term en een kleinere kans aan de termen waarbij we slechts één groep moeten testen.

6.3.3 Rendertijd

Het path tracing algoritme vereist over het algemeen een hogere rendertijd en een groter aantal intersectietesten. Enkel wanneer we de testscenes renderen met een laag aantal schaduwstralen is het path tracing algoritme sneller. Dit is te verklaren omdat we voor het occlusion map algoritme en het uitgebreide stochastische algoritme steeds acceleratie structuren moeten opbouwen voor de kandidaat blokkers. Enkel wanneer we voldoende schaduwstralen volgen is de kost voor het creëren van de acceleratiestructuren verwaarloosbaar ten opzichte van de tijdswinst door het gereduceerde aantal intersectietesten.

Verder zien we dat de rendertijden van het occlusion map algoritme zeer dicht bij de rendertijden van het uitgebreide stochastische algoritme liggen. Het uitgebreide stochastische algoritme is vaak echter trager, wat verklaard kan worden door het feit dat de kandidaat blokkers eerst gesorteerd moeten worden op basis van hun afstand tot het punt dat we willen belichten.

Daarnaast kunnen we ook opmerken dat de rendertijd voor het uitgebreide stochastische algoritme het laagst is wanneer we het merkwaardig product #1 gebruiken en het hoogst wanneer we de binomiaal formule gebruiken. De rendertijd hangt hierbij nauw samen met het aantal intersectietesten. Zoals uitgelegd in de vorige subsectie, is het aantal intersectietesten het hoogst wanneer we de binomiaal formule gebruiken.

Finaal zien we dat de rendertijd van het naïeve algoritme het grootst is voor alle testscene met uitzondering van de Sponza scene, waar enkel het path tracing algoritme nog slechter doet. Deze vaststelling is ook logisch, vermits het naïeve algoritme veel meer intersectietesten moet uitvoeren.

6.4 Besluit

We introduceerden het uitgebreide stochastische algoritme dat het naïeve algoritme verbetert met enkele uitbreidingen uit hoofdstuk 5. We analyseerden de performantie van het uitgebreide stochastische algoritme ten opzichte van het naïv algoritme, een path tracing algoritme en het occlusion map algoritme.

Uit de resultaten kunnen we besluiten dat het uitgebreide stochastische algoritme een verbetering is op het naïeve algoritme in rendertijd, aantal intersectietesten en MSE. Daarnaast zien we dat het occlusion map algoritme en het uitgebreide stochastische algoritme nu aan elkaar gewaagd zijn in MSE, rendertijd en aantal intersectietesten. Dit komt door de introductie van de Bounding Volume Hierarchie die zowel voor het occlusion map algoritme als het uitgebreide stochastische visibiliteits algoritme zeer efficiënt de groepen A en B kan testen op intersecties.

Verder kunnen we concluderen dat het path tracing algoritme een lagere MSE kan halen, maar dit vereist over het algemeen een hoger aantal intersectietesten en een langere rendertijd.

Hoofdstuk 7

Besluit

Dit hoofdstuk vat de voornaamste bevindingen uit deze thesis samen. In deze thesis hebben we voortgebouwd op de theorie van stochastische visibiliteit van Björn Engelen [4]. Stochastische visibiliteit is een nieuwe methode om de visibiliteit te evalueren, die het visibiliteitsproduct omvormt naar een som die stochastisch geschat wordt. In deze thesis hebben we gestreefd naar de integratie van deze theorie in een praktisch belichtingsalgoritme.

7.1 Het naïeve algoritme

Het naïeve algoritme was een eerste poging om tot een praktisch belichtingsalgoritme te komen. We integreerden hierbij de stochastische visibiliteitsevaluatie met een nieuwe datastructuur genaamd de occlusion map. De occlusion map laat ons toe om voor een punt in de scene de kandidaat blokkers voor dat punt te vinden. Vervolgens ontbinden we het visibiliteitsproduct van deze kandidaat blokkers in een som, waarbij we de kandidaat blokkers opdelen in twee groepen A en B.

Het naïeve algoritme heeft gemiddeld 26% minder intersectie testen nodig dan een equivalent algoritme dat de visibiliteit deterministisch evalueert. De kwaliteit van de afbeeldingen gerenderd met het naïeve algoritme is echter bijzonder slecht. De occlusion map slaagt er immers niet in om de fijne geometrie van de scene op te nemen in de datastructuur. Hierdoor zal die gemiste geometrie ook nooit getest worden op visibiliteit, waardoor de afbeelding nooit kan convergeren naar het juiste resultaat.

7.2 Het uitgebreide stochastische algoritme

Om de kwaliteit van de afbeeldingen te verhogen en de performantie van het naïeve algoritme te verbeteren, hebben we een aantal uitbreidingen toegevoegd aan het naïeve algoritme. De uitbreidingen die we hebben toegepast zijn:

- **Volumetrische occluders**: door de waterdichte modellen in de scene op te vullen met volumetrische occluders, compenseren we voor de gemiste geometrie

in de occlusion map.

- **Adaptieve occlusion map constructie**: we bouwen de occlusion map op een adaptieve manier, zodat we meer informatie over de kandidaat blokkers hebben in de zichtbare gebieden van de scene.
- **Analytische belichtingsintegratie**: we evalueren de belichting analytisch wanneer een punt volledig belicht wordt door een lichtbron. Deze situatie kunnen we detecteren met behulp van de occlusion map. Door de belichting analytisch te integreren, vermijden we een duur Monte Carlo integratie proces en berekenen we de belichting exact.
- **Integratie met acceleratiestructuren**: door acceleratiestructuren te bouwen over de groepen van de stochastische visibiliteit, kunnen we de visibiliteit veel sneller evalueren.
- **Importance sampling**: we passen importance sampling toe op de stochastische visibiliteitsevaluatie waardoor we de variantie reduceren.

7.3 Resultaten

Het uitgebreide stochastische algoritme is een competitief algoritme in vergelijking met het occlusion map algoritme dat de visibiliteit deterministisch test. Het uitgebreide stochastische algoritme vereist over het algemeen iets minder intersectietesten dan het occlusion map algoritme ten koste van een iets hogere fout in de afbeelding.

Het bekomen uitgebreide stochastische algoritme is geen verbetering op state of the art rendering algoritmen, maar het is wel een bewijs dat stochastische visibiliteit potentieel heeft om met een competitief algoritme te komen.

7.4 Verder onderzoek

Het concept van stochastische visibiliteit is nog een volledig onuitgewerkt gebied in computer graphics. Het uitgebreide stochastische algoritme, dat gebruik maakt van de occlusion map, is slechts een van de vele mogelijke manieren waarop stochastische visibiliteit geïntegreerd kan worden in een praktisch belichtingsalgoritme. Daarnaast zijn er nog veel meer mogelijkheden om het visibiliteitsproduct om te vormen in een som, die mogelijk een lagere variantie hebben.

Er zijn ook nog talrijke manieren waarop we de performantie van het uitgebreide stochastische algoritme kunnen verbeteren. Vermits we weten dat al de schaduwstralen eenzelfde punt als oorsprong hebben, zouden we speciale acceleratiestructuren kunnen bouwen over de groepen van kandidaat blokkers die deze eigenschap uitbuiten.

Daarnaast kan de occlusion map nog op talrijke manieren verbeterd worden. Wanneer een bepaalde regio veel identieke occlusion photonen bevat zouden we die regio als geconvergeerd kunnen beschouwen en de photonen in andere gebieden van de scene kunnen verdelen.

Alhoewel er nog veel onderzoek nodig is, hopen we met deze thesis aangetoond te hebben dat stochastische visibiliteit misschien een toekomst heeft in state of the art algoritmen.

Bijlagen

Bijlage A

Eurographics Paper

Eurographics Symposium on Rendering 2013
Nicolas Holzschuch and Szymon Rusinkiewicz
(Guest Editors)

Volume 32 (2013), Number 4

Probabilistic Visibility Evaluation for Direct Illumination

Niels Billen, Björn Engelen, Ares Lagae and Philip Dutré†

Department of Computer Science, KU Leuven, Belgium

Abstract

The efficient evaluation of visibility in a three-dimensional scene is a longstanding problem in computer graphics. Visibility evaluations come in many different forms: figuring out what object is visible in a pixel; determining whether a point is visible to a light source; or evaluating the mutual visibility between 2 surface points. This paper provides a new, experimental view on visibility, based on a probabilistic evaluation of the visibility function. Instead of checking the visibility against all possible intervening geometry, the visibility between 2 points is now evaluated by testing only a random subset of objects. The result is not a Boolean value that is either 0 or 1, but a numerical value that can even be negative. Because we use the visibility evaluation as part of the integrand in illumination computations, the probabilistic evaluation of visibility becomes part of the Monte Carlo procedure of estimating the illumination integral, and results in an unbiased computation of illumination values in the scene. Moreover, the number of intersections tests for any given ray is decreased, since only a random selection of geometric primitives is tested. Although probabilistic visibility is an experimental and new idea, we present a practical algorithm for direct illumination that uses the probabilistic nature of visibility evaluations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

The evaluation of visibility in a three-dimensional scene has a long history in the field of computer graphics. Early algorithms focused mainly on the spatial relationship between geometric entities, and were closely related to techniques derived from descriptive geometry or technical drawings. This research led to many different techniques that involved line- and polygon clipping and point-in-polygon tests . With the introduction of the framebuffer and later the Z-buffer, visibility became discretized. Instead of performing continuous operations on lines and polygons, the focus now lay on operations in the image plane consisting of pixels. While many of the visibility operations were designed to determine what is visible as seen from the camera, the visibility problem using the position of the light sources (determining shadow regions) spawned its own collection of algorithms. Again, this led to continuous operations such as the computation of shadow volumes, shadow polygons, etc.; as well as to discretized data-structures such as the shadow map. A full survey of these techniques is outside the scope of this paper, but the interested reader can find a lot of information about the development of visibility calculations in general computer graphics books, e.g. [FvDFH95, AMHH08, ESAW11].

With the advent of ray tracing, two specific visibility evaluations became more prominent. For determining the visibility from the camera through a pixel, one is interested in the first visible surface point along a viewing ray. This type of visibility query is also useful when tracing recursive rays (reflective and refractive rays, or more generally, indirect rays in stochastic ray tracing). For shadow calculations, tracing a ray between a point to be shaded and a light source, a slightly different type of evaluation is needed. Now, one is not interested in the first intersection point encountered, but whether an intersection with surrounding geometry is present at all. The efficiency of both types of queries is speeded up significantly by the use of proper acceleration structures (e.g., [Wal07, WMG*08]). Most acceleration schemes follow the ray from start to end, moving through a number of spatial grid cells or hierarchical bounding boxes, each containing a number of geometric primitives. Once an

† e-mail: philip.dutre@cs.kuleuven.be

intersection is found, any primitives positioned farther along the ray do not need to be considered any further.

In the context of global illumination algorithms, the visibility evaluation between two points exchanging light energy (e.g. a point to be shaded and a point located at the surface of a light source), is written as part of the integrand of the illumination integral [DBB06, PH10]. These integrals are often evaluated using Monte Carlo integration, sampling a number of rays in the total integration domain [RAMN12]. Well-known and tested optimizations include importance sampling, in which either a BRDF-kernel or an incoming radiance distribution is sampled proportionally. The visibility function, however, is almost always evaluated exactly. Only a limited number of approaches have explore non-traditional evaluation of visibility [HDG99, DSDD07].

In this paper we introduce a new and experimental view on visibility evaluations, which considers the visibility estimation as a numerical part of the Monte Carlo estimation of the illumination integral. This stochastic evaluation of visibility is performed by testing only a random subset of the geometry likely to be located between two surface points. We will show that — perhaps somewhat counterintuitive — such an evaluation produces unbiased images. Since this technique reduces the number of intersection tests for each ray, efficiency can be gained. However, this process also introduces a new source of stochastic noise in the image. Therefore, a careful balance between noise and efficiency has to be found. By exploring this new approach on visibility, we enlarge the toolbox of methods available that can be used to evaluate visibility in graphics algorithms.

In summary, the contributions of this paper are:

- We develop a theoretical framework to evaluate visibility on stochastic subsets of geometric primitives. Our proposed approach considers visibility as a numerical function that can be sampled stochastically, and produces unbiased images in the context of global illumination algorithms.
- We propose a practical algorithm that uses the notion of stochastic visibility for computing direct illumination in a scene, based on the idea of an occlusion map. For each shadow ray to be processed, a random subset of the geometry most likely to intersect the shadow ray is tested for intersection. We show that we can reach the same image quality while performing fewer intersections tests
- We provide some additional theoretical insights in how the visibility function can be evaluated in the context of graphics algorithms.

2. Theoretical framework

The visibility function $V(x,y)$ in direct illumination algorithms is usually evaluated between 2 surface points x and y [DBB06]. If x and y are mutually visible, then $V(x,y) = 1$.

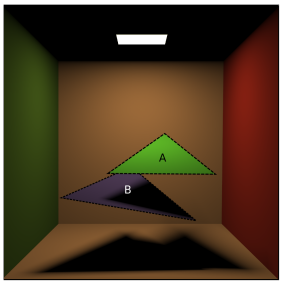


Figure 1: Box containing two candidate blockers A and B. Note that the walls, ceiling and floor are not considered for intersection testing.

Otherwise, $V(x,y) = 0$. In order to evaluate $V(x,y)$, all candidate geometric primitives that are located between x and y need to be checked for intersection. A proper acceleration structure can limit the list of possible intersecting geometry, or is able to process them in sorted order between x and y.

The set of candidate geometric primitives that could intersect the line segment xy can be written as $Z = \{z_1, z_2, z_3, ..., z_n\}$. For now, we make abstraction of how this set is determined, and merely assume that the geometric primitives in Z are the ones that need to be tested explicitly for intersection. The total visibility $V(x,y)$ can then be written as the product of individual visibility evaluations:

$$V(x,y) = V_{z_1}(x,y) \cdot V_{z_2}(x,y) ... V_{z_n}(x,y) = \prod_{i=1}^{n} V_{z_i}(x,y) \quad (1)$$

$V_{z_i}(x,y)$ represents the visibility value (0 or 1) of the line segment xy, only taking into account the possible intersection with primitive z_i. The product represents the total visibility against the entire set of candidate blocking geometry. If at least one geometric primitive from the set results in a visibility value of 0, the total visibility equals 0.

It is important to realize that we will only concern ourselves with the numerical evaluation of visibility (0 or 1), as needed for direct illumination, and not with visibility in the sense of trying to determine what surface is visible along a given (viewing) ray. We will also refer to geometric primitives that need to be tested for intersection as candidate blockers or potential blockers.

2.1. Stochastic evaluation of visibility

To explain the core idea of probabilistic visibility, let us simplify Eqn. 1 by only considering two candidate intersecting blockers, A and B (Fig. 1). The visibility term $V(x,y)$ equals $V_A(x,y) \cdot V_B(x,y)$, in which V_A represents the visibility checking only against blocker A, and V_B checks only against blocker B.

The key insight of our procedure is to rewrite this product as a sum, and stochastically select only one of the terms of the sum to evaluate the overall function. A sum $S = s_1 + s_2 +$

Exact values			Stochastic evaluation			Var
V_A	V_B	$V_A.V_B$	$3V_A$	$3V_B$	$3(\overline{V_A}\overline{V_B}-1)$	-
0	0	0	0	0	0	0
0	1	0	0	3	-3	6
1	0	0	3	0	-3	6
1	1	1	3	3	-3	8

Table 1: Stochastic visibility evaluation for two polygons A and B. Each line shows one of 4 possible visibility configurations. Exact visibility values (leftmost columns), stochastic value for each of the 3 terms that can be selected (middle columns), variance of the stochastic process (right column).

$\ldots + s_n$ can be stochastically estimated by selecting a single term s_i with probability p_i. Provided all $p_i > 0$ and their sum equals 1, $\tilde{S} = s_i/p_i$ is then an unbiased estimator for S. One of the most simple decomposition of a product into a sum of terms is based on the algebraic product $ab = a+b+(1-a)(1-b)-1$. Using complement notation ($\overline{a} = 1-a$), Eqn. 1 for 2 blockers is then written as:

$$\begin{aligned} V(x,y) &= V_A(x,y) \cdot V_B(x,y) \\ &= V_A(x,y) + V_B(x,y) + (\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - 1) \end{aligned} \quad (2)$$

By stochastically selecting one of the 3 terms, each with probability p_1, p_2 and p_3 ($p_1+p_2+p_3 = 1$), we obtain the following evaluation scheme:

$$\tilde{V}(x,y) = \begin{cases} \frac{V_A(x,y)}{p_1} & \text{with probability } p_1 \\ \frac{V_B(x,y)}{p_2} & \text{with probability } p_2 \\ \frac{\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - 1}{p_3} & \text{with probability } p_3 \end{cases} \quad (3)$$

The numerical value of the total visibility $V(x,y)$ is therefore estimated, *in an unbiased manner*, by only intersecting with polygon A in the first case, only intersecting with polygon B in the second case, or intersecting *both* in the third case. Note that, depending on the choices of $p1$, $p2$ and $p3$, visibility values can be larger than 1. The third term can even become negative. As an example, assume $p1$, $p2$ and $p3$ are all equal to $1/3$, leading to the stochastic evaluations listed in Tab. 1. For each of 4 possible visibility configurations, the exact values are listed, along with the stochastic evaluation when each of the 3 possible terms would be selected. The last column lists the exact value of variance (computed analytically) of the stochastic process. It is straightforward to check that the expected value of this stochastic process yields the correct visibility value in each of the 4 visibility configurations. It might be counterintuitive to deal with visibility values different from 0 or 1, but since we are interpreting visibility as a numerical value rather than a geometric property, this is not unsurprising.

This numerical interpretation of visibility can be exploited when including $V(x,y)$ in the illumination integral for direct illumination calculations [DBB06]. Radiance L at a surface point x is written as (without loss of generality, we assume a diffuse surface):

$$L(x) = \int_S f_r(x) L(y \to x) V(x,y) G(x,y) dS_y \quad (4)$$

with S the area of the light source, y being a surface point on the light source, f_r the BRDF, and $G(x,y)$ the geometric coupling term containing two cosine factors and the inverse of the distance squared between x and y. Splitting the visibility term as above:

$$\begin{aligned} L(x) = &\int_S f_r(x) L(y \to x) V_A(x,y) G(x,y) dS_y \\ &+ \int_S f_r(x) L(y \to x) V_B(x,y) G(x,y) dS_y \\ &+ \int_S f_r(x) L(y \to x) (\overline{V_A(x,y)} \cdot \overline{V_B(x,y)} - 1) G(x,y) dS_y \end{aligned} \quad (5)$$

The evaluation of $L(x)$ can now be considered as evaluating the above sum of integrals, by stochastically selecting only one of the 3 terms for each shadow ray cast towards the light source, resulting in an unbiased estimator for $L(x)$.

The first row of images shown in Fig. 2 are rendered using this stochastic evaluation of visibility, for a gradually increasing number of shadow rays per pixel. Notice that the area where the shadows of both polygons overlap is rendered correctly irrespective of the number of shadow rays. This is also obvious from Tab. 1, since in this case, the numerical variance equals 0. The areas in which only a single shadow is present do contain noise, since for one-third of the shadow rays, the relevant polygon will not be checked for visibility. The areas without any shadows at all do exhibit dark noise as well, since their visibility values are the result of adding positive and negative values to converge to an average value of 1 (see Tab. 1).

The overall advantage is that for each shadow ray, on average, only $(1+1+2)/3 = 1.33$ polygons need to be checked for intersection, versus 2 polygons for a regular visibility evaluation. Although we expect an increase in image noise, there is room for a trade-off between speed (number of intersections per shadow ray) and image quality (noise).

2.2. Alternative decompositions

We can expect that other decompositions of the visibility product into a sum can lead to less noise in the final direct illumination image. An interesting choice is to distribute the -1 term evenly over all 3 visibility terms. With p_1, p_2 and p_3 again all equal to $1/3$, the different possibilities are listed in Tab. 2.

Again, it is easy to check that the expected value in each of the 4 configurations equals the correct visibility value. Although the variance generally is lower in value compared to the previous evaluation in Tab. 1, we have introduced variance for areas which are covered by both shadows, and lowered the variance in completely visible areas or areas only covered by one of both shadows. This is an expected result,

Exact values			Stochastic evaluation			Var
V_A	V_B	$V_A.V_B$	$3V_A-1$	$3V_B-1$	$3\overline{V_AV_B}-1$	-
0	0	0	-1	-1	2	2
0	1	0	-1	2	-1	2
1	0	0	2	-1	-1	2
1	1	1	2	2	-1	2

Table 2: Stochastic visibility evaluation for two polygons A and B with equal distribution of the -1 term, each line showing one of 4 possible visibility configurations. Exact visibility values (leftmost columns), stochastic value for each of the 3 terms that can be selected (middle columns), variance of the stochastic process (right column).

Exact values			Stochastic evaluation			Var
V_A	V_B	$V_A.V_B$	$-\frac{3V_A}{254}$	$-\frac{3V_B}{254}$	$\frac{3(V_A+V_B)^8}{254}$	-
0	0	0	0	0	0	0
0	1	0	0	-3/254	3/254	4.65e-5
1	0	0	-3/254	0	3/254	4.65e-5
1	1	1	-3/254	-3/254	768/254	2.048

Table 3: Stochastic visibility evaluation for two polygons A and B using binomial decomposition, each line showing one of 4 possible visibility configurations. Exact visibility values (leftmost columns), stochastic value for each of the 3 terms that can be selected (middle columns), variance of the stochastic process (right column).

since we are effectively always evaluating the constant -1 term present in the sum decomposition, whereas before, we only evaluated it for $1/3$ of the shadow rays. Fig. 2, second row, shows images rendered with this decomposition. We see noise in the completely shadowed area of the image, but compared to the previous case, visible noise in other areas of the image has decreased. One can observe an overall faster convergence towards the exact solution.

We also experimented with a decomposition of the visibility function based on the binomial theorem. Since V_A and V_B equal either 0 or 1, we can write:

$$\begin{aligned}(V_A(x,y)+V_B(x,y))^n =& V_A(x,y)+V_B(x,y)\\ &+(2^n-2)V_A(x,y)V_B(x,y)\end{aligned} \tag{6}$$

and thus:

$$\begin{aligned}V(x,y) =& -\frac{V_A(x,y)}{2^n-2}-\frac{V_B(x,y)}{2^n-2}\\ &+\frac{(V_A(x,y)+V_B(x,y))^n}{2^n-2}\end{aligned} \tag{7}$$

Choosing $n=8$, and selecting each of 3 possible terms with probability $1/3$ provides the values listed in Tab. 3. The variance all shadowed regions is now very small, but variance is present in completely illuminated regions.

The resulting images are shown in the third row of Fig. 2. We have no variance in areas where both polygons cast a shadow. Noise in shadowed areas caused by a single polygon is far less as in both previous cases, as is predicted by the variance values in Tab. 3. The illuminated areas though, show quite some noise for a low number of shadow rays. In this case, a pixel might end up will a small negative visibility value (subsequently capped to 0). Experimenting with larger values for n in the binomial decomposition did not produce significantly different results.

2.3. Further discussion

Multiple blockers. So far, we have considered the case for 2 blocking polygons. The theory still holds when we have a larger set of potentially intersecting geometry. By subdividing the set in 2 groups A and B, the above equations still hold. A visibility evaluation for a group then involves explicitly checking all polygons in that group for intersection. A recursive further subdivision of a selected group can also be considered, but various tests showed that this option increases noise significantly in the image. This is not surprising, since reducing visibility tests for a large set of potential blockers to only a single test vs. a single blocker, is unlikely to provide accurate (although unbiased) results. For a numerical insight, consider the values in Tab. 1. Stochastic visibility values for a single shadow ray can be either 0, 3 or -3. Averaging a large number of samples either produces a value of 0 or 1. If groups are recursively subdivided, the stochastic visibility values are multiplied by an additional factor of 3 for each recursive subdivision, resulting in very large positive and negative values, that still need to average to 0 or 1. Such a process with increasingly larger values inherently increases variance. This increase in variance is also noticeable when we use an acceleration structure such as a regular grid to process the shadow ray. Each cell the ray passes through contains a number of polygons, whose visibility is evaluated using the procedures above. However, by stacking up all stochastic evaluations for all cells the ray passes through, we reach visibility estimates which have high absolute values, but still need to average out to 0 or 1.

Negative visibility values. It is possible that due to negative visibility values, the total illumination integral for a surface point evaluates to a negative value, especially for points for which the exact visibility value equals 0. Although the variance decreases with a growing number of shadow rays, there still is a significant probability that the average visibility value will not end up being exactly 0 due to the averaging of positive and negative evaluations. In our implementation, we clamp all pixels with negative values to 0. This makes the image a little bit too bright on average, but this can be corrected for by accumulating all negative light in pixels and distributing it equally over (neighboring) pixels. This is a similar approach as used in the redistribution of unshot energy in radiosity algorithms [CCWG88].

Similarity to BRDF decomposition. The decomposition of the visibility term in a sum of terms, thereby splitting the illumination integral in a sum of integrals (Eqn. 5), is quite similar to the evaluation of the indirect illumination integral

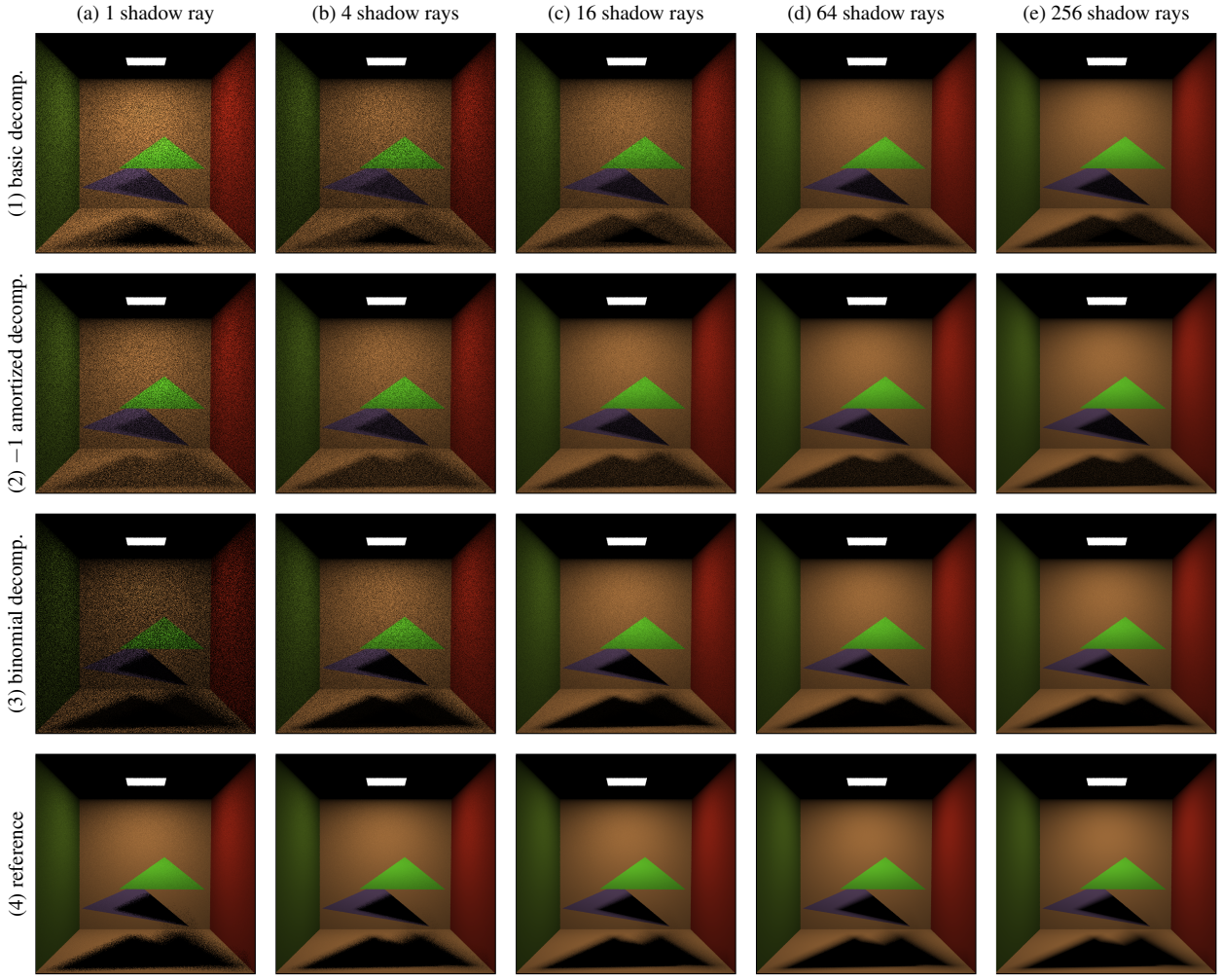


Figure 2: Stochastic visibility evaluation for different decomposition of the visibility function (rows 1-3) and for a gradually increasing number of shadow rays per pixel (columns a-e) for the scene shown in Fig. 1. A reference image using deterministic visibility evaluation is shown in row 4.

in global illumination, in which the BRDF is split in a sum of a diffuse and a specular term. An indirect illumination ray is then chosen with discrete probability as being reflected either through the diffuse or the specular part of the BRDF, and the resulting evaluation is an estimate for the reflection behavior of the complete BRDF.

3. Practical algorithm using occlusion mapping

3.1. Design decisions and algorithm description

To use probabilistic visibility in a direct illumination algorithm, we need to have knowledge of the potential blockers between a surface point to be shaded and the light source(s) in the scene. We need to consider the entire group of blockers that can possibly intersect the ray as a whole, and split this group in two subgroups. Ideally, we would like to use an acceleration structure that quickly selects potential blockers based on their geometric proximity along the entire length of the shadow ray. Traditional acceleration structures such as spatial grids or hierarchical bounding volumes are not immediately suitable since they are specialized in finding the first blocker along a ray by proceeding through a number of spatial cells the ray passes through. As mentioned before, splitting up the visibility evaluation by subsequently evaluating visibility one cell at a time, yields a negative effect on the variance of the visibility evaluation.

For our purposes, directional structures such as a light-buffer [HG86] or a 5D ray hierarchy [AK87] are much better suited. These structures are able to select all blockers located in a narrow directional shaft around the entire length of the ray. When processing a shadow ray, we can then retrieve the blockers that have an effect on the total visibility function

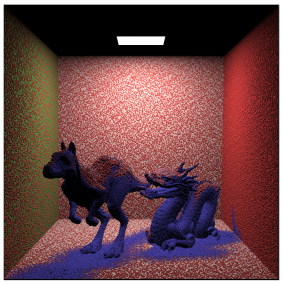


Figure 3: Location of light photons (red) and occlusion photons (blue) for the scene shown in Fig. 7(b).

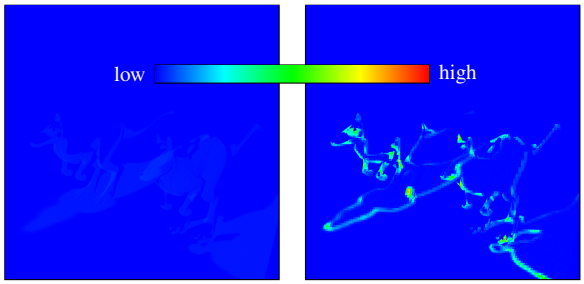
for the ray, and proceed with that group for our stochastic evaluation.

As a directional structure that selects blockers based on ray direction, we employ an *occlusion map*. Our occlusion map is closely related to shadow photons, introduced in the context of photon mapping [Jen01]. The occlusion map stores information about all blockers located between an occlusion photon (a surface point) and the light source. The location of occlusion photons tells us something about the location of shadows in the scene, as well as about the geometry casting these shadows. We will use the occlusion map to discriminate between umbra, penumbra, and illuminated regions. We will only use probabilistic visibility for the penumbra regions.

Our direct illumination algorithm is composed of the following steps:

1. Construction of the occlusion map, by storing information at surface points (*occlusion photons*) regarding blockers between that point and the light source (Sec. 3.2). If no blockers are present, we consider the point as a *light photon*.
2. During rendering, for each pixel a viewing ray is traced through the scene. A lookup in the occlusion map retrieves nearby occlusion and light photons for the visible point through the pixel (Sec. 3.3). The following possibilities occur:
   a. No occlusion photons are found: the light source is deemed to be completely visible and the illumination will be evaluated analytically.
   b. Only occlusion photons are found: the point to be shaded is considered to be located in the umbra region and nu further illumination computations are made.
   c. A mix of occlusion photons and light photons is detected: the point is likely to be situated in a penumbra region w.r.t. the light source. The potential blockers from the occlusion photons are gathered and split in two groups A and B for probabilistic visibility evaluation.

We will now describe each of these steps in more detail.



(a) non-adaptive (b) adaptive

Figure 4: Occlusion photon density for (a) non-adaptive and (b) adaptive generation of occlusion photons for the scene shown in Fig. 7(a).

3.2. Construction of the occlusion map

We need to be able to quickly identify the set of candidate blockers between a point to be shaded and a light source. Besides creating light photons in the regions illuminated by a light source, we also create occlusion photons in the regions invisible to the light source. These occlusion photons store the set of all occluders between the position of the occlusion photon and the light source. Since occlusion photons are only present in shadowed regions, the location of occlusion photons is an indication of the presence of a shadow in that particular part of the scene.

The occlusion map can be constructed in a couple of different ways. A straightforward approach distributes photons from the light source, and a light photon is created at the first intersection point visible from the light source. The photon ray is then traced further through intersecting geometry and occlusion photons are created at all subsequent intersection points. All previously intersected geometric elements are stored with the occlusion photon. Since we will only trace shadow rays in the penumbra regions where a mix of light photons and occlusion photons are detected, we would like to distribute more occlusion photons in these penumbra regions. This is difficult to achieve while distributing the photons from the light source.

We therefore opted for a camera-driven generation of occlusion photons. A first batch of viewing rays (we used 200K rays for our images) is uniformly distributed over the image plane. A shadow ray is traced from each of those 200K intersection points. If the light source is reached unhindered, a light photon is created. Otherwise, an occlusion photon which stores all the blockers encountered along the shadow ray is stored in the occlusion map (see Fig. 3). For each group of 8x8 pixels in the image plane, the number of occlusion and light photons are compared. Subsequent batches of viewing rays (again, 200K each), are distributed according to a density distribution that favors previously detected penumbra regions (see Fig. 4). The net result is that relatively more occlusion photons are generated in penumbra regions.

3.3. Rendering phase

The rendering phase generates a viewing ray through each pixel, finding the first visible point x. In order to compute the illumination at x, the corresponding occlusion map is queried to locate the nearest 100 occlusion photons within a maximum set distance. As explained before, the following cases are considered:

- If no occlusion photons are found, x is assumed to be completely illuminated by the light source, and the illumination is computed analytically [Arv95]. This currently limits our technique to diffuse surfaces only.
- If occlusion photons are detected, we search for the closest 100 light photons, again within a set maximum distance. If no light photons are present, we assume x is in the umbra region, and no further computations are made. Otherwise, we assume x is located within the penumbra.

In the last case, we collect all unique blockers that are stored in the 100 closest occlusion photons. These blockers are considered the most likely candidates to intersect the shadow ray. Note that this set is not conservative, i.e. it is possible that a blocker intersecting a shadow ray starting at x will not be present in this set. This can happen if this particular blocker was not detected during the construction of the occlusion map. Especially with very small blockers relative to the density of the occlusion map, this might be a problem.

We subsequently subdivide the blockers in two groups, and then proceed with probabilistic visibility evaluation using Eqn. 7. To subdivide all blockers in 2 groups A and B, they are sorted according to their subtended solid angle as seen from x. Blockers are subsequently put into one of both groups, keeping the summed solid angle of both groups roughly equal. The rationale behind this split is that we will initially attribute equal probabilities to A and B during probabilistic visibility evaluations. If they both subtend equal solid angle, chances are good they both are as likely to intersect the shadow ray.

Note that the occlusion map can also be used to evaluate visibility deterministically by intersecting *all* blockers stored in the occlusion photons. In our algorithm, we merely employ the occlusion map to quickly select a number of candidate blockers to serve as input for the probabilistic visibility evaluation. Discussing advantages and limitations of the occlusion map without probabilistic visibility evaluation falls outside the scope of this paper.

3.4. Efficiency and implementation issues

Importance sampling of visibility terms. To improve performance, importance sampling for the three different visibility terms of Eqn. 7 is implemented (see Fig. 5). Initially, all probabilities are set to $1/3$. After the first group of 64 shadow rays is evaluated, the probabilities are refined based on the hit ratios of the shadow rays versus groups A and B.

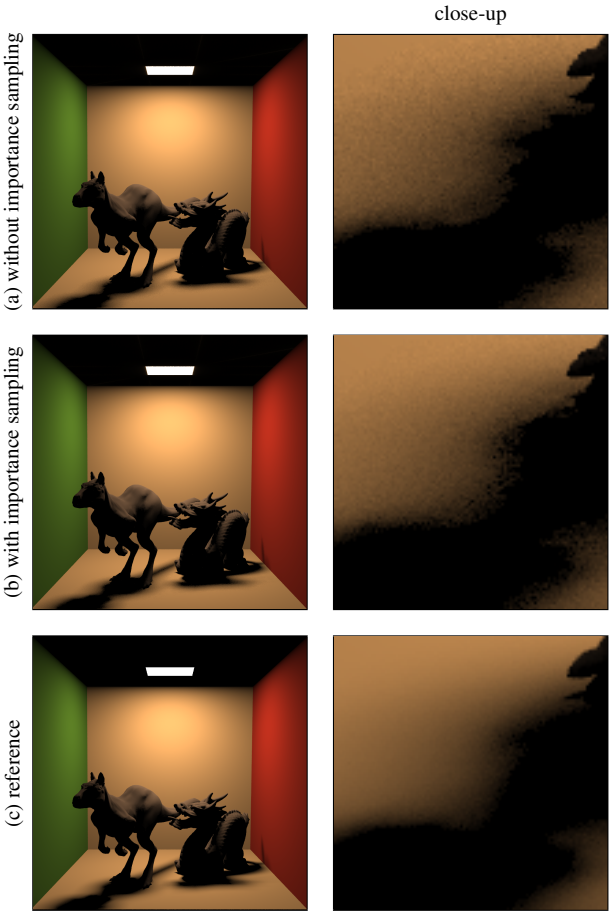
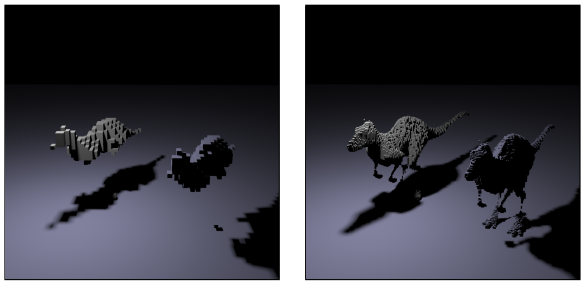


Figure 5: Importance sampling of visibility terms. Stochastic visibility evaluation (a) without and (b) with importance sampling of visibility terms for the scene shown in Fig. 7(b). A reference using deterministic visibility evaluation (c) and close-ups are also shown.

This allows us to assign probabilities to the different visibility terms proportional to their actual numerical values, and therefore lower the variance of the stochastic visibility evaluations.

Volumetric occluders. Probabilistic visibility performs better if the selection of candidate blockers is accurate. For scenes with a large number of small polygons, such small polygons might be missed entirely in the occlusion map, and therefore not show up in the visibility evaluations. Volumetric occluders can be used to fill up solid objects with a solid octree [DKH09] (Fig. 6). If these occluders are also stored as part of the occlusion map, the chance of missing visibility events is largely reduced.



(a) depth 4 (b) depth 6

Figure 6: Visualization of the volumetric occluder tree at (a) depth 4 and (b) depth 6 for the scene shown in Fig. 7(a). Volumetric occluders reduce missed visibility events and enhance the reliability of probabilistic visibility.

4. Discussion

4.1. Results

We evaluated stochastic visibility evaluation with occlusion mapping on different scenes (Fig. 7). To test the limits of our algorithm, we included some test scenes with very challenging penumbra shadow patterns (e.g. a Menger grid and the Yeah Right model). For solid objects, volumetric occluders were used as well.

Tab. 4 illustrates the performance of our algorithm for the scenes in Fig. 7. Both execution times and number of intersection tests are listed. Since we want to investigate the isolated effect of probabilistic visibility, we compared two rendering approaches, both using the occlusion map, but with and without the use of probabilistic visibility. Our stochastic visibility estimation reduces the number of intersections for an equal number of shadow rays on average by 22% while decreasing the rendering time on average by 14%. This is due to the probabilistic evaluation of visibility in detected penumbra regions. The corresponding images indicate equal visual quality.

When comparing reference images with images rendered with the occlusion map, we notice that shadow regions tend to become smaller. This is due to the difficulty of finding all potential blockers in the penumbra regions. This is clearly seen in the scene with the Yeah Right model where very fine shadows tend to disappear. However, regions which are categorized as either in umbra or completely illuminated are always fully converged due to analytic integration of the illumination.

4.2. Limitations

The proposed algorithm using the occlusion map and the probabilistic evaluation of visibility has a number of limitations:

- With an exact evaluation of the visibility, noise due to visibility is only present in penumbra regions. Probabilistic visibility introduces noise in umbra and illuminated regions as well. However, the number of intersections per shadow ray is reduced. Further research might be able to exploit this trade-off between speed and quality in a more fundamental way.
- The occlusion map is an acceleration structure designed to quickly select potential blockers for a given surface point. The occlusion map itself is constructed stochastically, hence introducing some additional sources of error (small polygons might be missed completely). The use of volumetric occluder alleviates these missed visibility events.
- Especially for large amounts of geometry present in the scene, it remains doubtful whether probabilistic visibility can compare favorably with more traditional, often very efficient, accelerations structures. Hierarchical grids are able to find an intersection very quickly, often after only a handful of intersection tests. Future work on the use and exploitation of probabilistic visibility in conjunction with acceleration structures might therefore be interesting.

5. Conclusion

We have presented a new and experimental view on the evaluation of visibility, based on a probabilistic evaluation of various visibility terms. By integrating this numerical evaluation into the illumination integral for direct illumination, we are able to generate unbiased images. Although the number of intersection tests per shadow ray are reduced, additional noise is introduced into the image. A careful balance is therefore needed between computation time and image quality.

A practical algorithm using stochastic visibility should use a directional acceleration scheme, since potential occluders located close to the shadow ray should be selected quickly. We have shown that using an occlusion map, encoding visibility events in a large set of occlusion photons, is suited for this task. Using various optimizations, a competitive algorithm has been constructed.

We hope that by having presented a novel evaluation of the visibility function, new types of visibility algorithms can be constructed, and additional new insights may be gained by using probabilistic visibility.

Acknowledgements

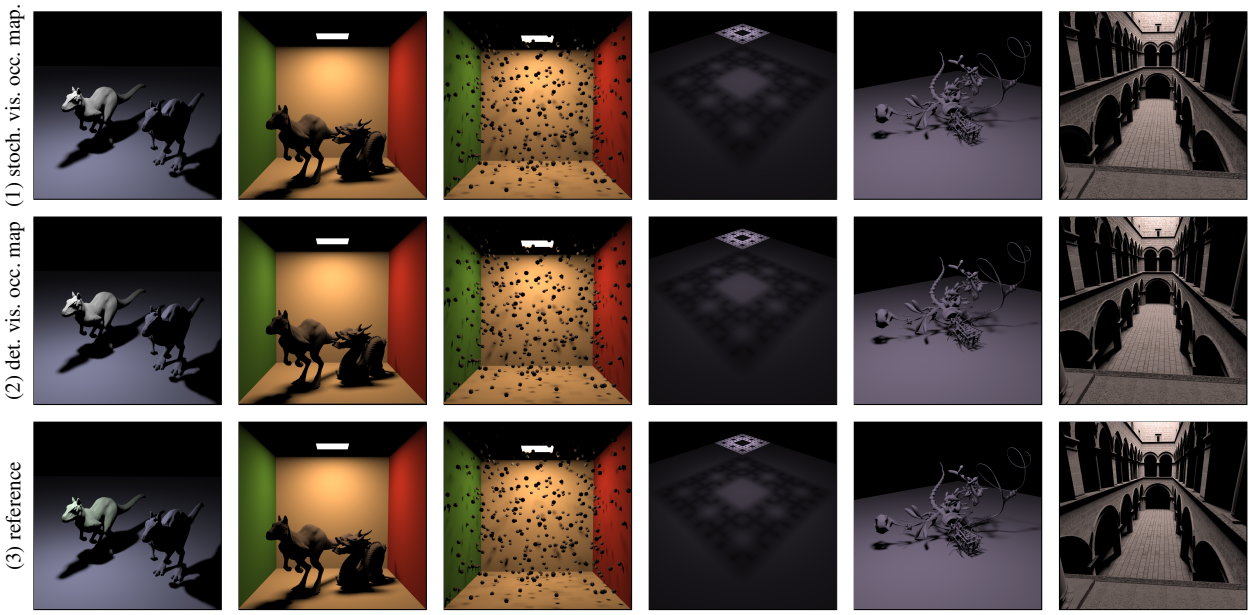
The scenes used in this paper are courtesy of PBRT [PH10]. We would like to thank the anonymous reviewers. Ares Lagae is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

References

[AK87] ARVO J., KIRK D.: Fast ray tracing by ray classification. In *Computer Graphics* (1987), vol. 21, pp. 55–64. 5

scene	time (in sec)		intersections (in millions)	
	occlusion map	prob.vis.	occlusion map	prob.vis.
Killeroo	306s	262s (-14.4%)	27494M	21047M (-23.4%)
Cornell w. Dragon and Killeroo	990s	827s (-16.4%)	96453M	76283M (-20.9%)
Icosahedrons	714s	533s (-25.4%)	61561	37913 (-38.2%)
Menger cube	637s	578s (-9.3%)	63470M	51927M (-18.2%)
Yeah Right	663s	560s (-15.5%)	70392M	55203M (-21.5%)
Sponza	628s	588s (-6.3%)	58449M	44867M (-23.2%)

Table 4: Performance of stochastic visibility evaluation.



(a) Killeroos (b) Cornell w. Dragon (c) Icosahedron (d) Menger (e) YeahRight (f) Sponza

Figure 7: Stochastic visibility renderings with the occlusion map for different scenes (a-f). For comparison purposes, deterministic visibility evaluation using the occlusion map (row 2) and a reference image using deterministic visibility evaluation (row 3) are also shown.

[AMHH08] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-Time Rendering 3rd Edition*, 3rd ed. A K Peters/CRC Press, 2008. 1

[Arv95] ARVO J.: *Analytic methods for simulated light transport*. PhD thesis, Yale University, 1995. 7

[CCWG88] COHEN M. F., CHEN S. E., WALLACE J. R., GREENBERG D. P.: A progressive refinement approach to fast radiosity image generation. In *Computer Graphics* (1988), vol. 22, pp. 75–84. 4

[DBB06] DUTRÉ P., BALA K., BEKAERT P.: *Advanced Global Illumination*, 2nd ed. A K Peters/CRC Press, 2006. 2, 3

[DKH09] DJEU P., KEELY S., HUNT W.: Accelerating shadow rays using volumetric occluders and modified kd-tree traversal. In *Proceedings of the Conference on High Performance Graphics 2009* (2009), pp. 69–76. 7

[DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics 26*, 3 (2007), 61:1–61:10. 2

[ESAW11] EISEMANN E., SCHWARZ M., ASSARSSON U., WIMMER M.: *Real-Time Shadows*. A K Peters/CRC Press, 2011. 1

[FvDFH95] FOLEY J. D., VAN DAM A., FEINER S. K., HUGHES J. F.: *Computer graphics: principles and practice*, 2nd ed. Addison-Wesley Professional, 1995. 1

[HDG99] HART D., DUTRÉ P., GREENBERG D. P.: Direct illumination with lazy visibility evaluation. In *Proceedings of SIGGRAPH 99* (1999), pp. 147–154. 2

[HG86] HAINES E. A., GREENBERG D. P.: The light buffer: A shadow-testing accelerator. *IEEE Computer Graphics and Applications 6*, 9 (1986), 6–16. 5

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*, 2nd revised ed. A K Peters/CRC Press, 2001. 6

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering*, 2nd ed. Morgan Kaufmann, 2010. 2, 8

[RAMN12] RAMAMOORTHI R., ANDERSON J., MEYER M., NOWROUZEZAHRAI D.: A theory of monte carlo visibility sampling. *ACM Transactions on Graphics 31*, 5 (2012), 121:1–121:16. 2

[Wal07] WALD I.: On fast construction of sah-based bounding volume hierarchies. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (2007), pp. 33–40. 1

[WMG*08] WALD I., MARK W. R., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S. G., SHIRLEY P.: State of the art in ray tracing animated scenes. *Computer Graphics Forum 28*, 6 (2008), 1691–1722. 1

Bijlage B

Poster

Stochastische Visibiliteit in Rendering Algoritmen a.d.h.v. de occlusion map

KATHOLIEKE UNIVERSITEIT LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Master Computer-wetenschappen

Masterproef
Niels Billen

Promotor
Prof. dr. ir. Philip Dutré

Academiejaar 2012-2013

Situering en doelstelling

- De visibiliteit $V(x, y)$ tussen twee punten x en y is een driedimensionale ruimte is een veel voorkomende, maar rekenintensieve operatie.
- We splitsen de kandidaat blokkers tussen x en y in twee groepen A en B en evalueren de visibiliteit stochastisch met behulp van de volgende ontbinding:

$$V(x,y) = \begin{cases} \dfrac{V_A(x,y)}{p_1} & met\ kans\ p_1 \\ \dfrac{V_B(x,y)}{p_2} & me\ kans\ p_2 \\ \dfrac{\overline{V_A(x,y)V_B(x,y)}}{p_3} & met\ kans\ p_3 \end{cases}$$

Occlusion map

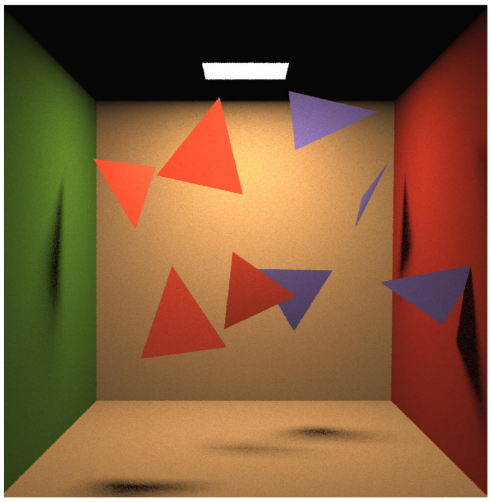
- Preprocess waarbij photonen vanaf de lichtbron doorheen de ruimte verdeeld worden
  - Licht photonen op belichte posities
  - Occlusion photonen in schaduw posities
- Occlusion photonen onthouden de geometrische primitieven tussen het photon en de lichtbron.
- Blokkers gevonden in de occlusion photonen rond een punt x worden gebruikt om de groepen A en B te creëren.
- Laat toe onderscheid te maken tussen volledig belichte gebieden, volledig beschaduwde gebieden en zachte schaduw gebieden.

Onderzochte topics

- Variantie reductie door importance sampling
- Integratie met acceleratie structuren
- Meer blokkers introduceren met behulp van volumetrische occluders.

Resultaten

- Vergelijkbare kwaliteit en performantie met deterministische visibiliteit.



Bibliografie

[1] J. R. Arvo. Analytic methods for simulated light transport. Technical report, 1995.

[2] P. Djeu, S. Keely, and W. Hunt. Accelerating shadow rays using volumetric occluders and modified kd-tree traversal. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 69–76, New York, NY, USA, 2009. ACM.

[3] P. Dutre, K. Bala, P. Bekaert, and P. Shirley. *Advanced Global Illumination*. AK Peters Ltd, 2006.

[4] B. Engelen. Stochastic visibility in rendering algorithms. Master's thesis, KU Leuven, 2012.

[5] H. W. Jensen. *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.

[6] M. H. Kalos and P. A. Whitlock. *Monte Carlo methods. Vol. 1: basics*. Wiley-Interscience, New York, NY, USA, 1986.

[7] M. Pharr and G. Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[8] I. Wald, W. R. Mark, J. Günther, S. Boulos, T. Ize, W. Hunt, S. G. Parker, and P. Shirley. State of the art in ray tracing animated scenes. In D. Schmalstieg and J. Bittner, editors, *STAR Proceedings of Eurographics 2007*, pages 89–116. The Eurographics Association, Sept. 2007.

Fiche masterproef

Student: Niels Billen

Titel: Stochastische Visibiliteit in Rendering Algoritmen a.d.h.v. de occlusion map

Engelse titel: Stochastic Visibility in Rendering Algorithms using the occlusion map

UDC: 621.3

Korte inhoud:
In deze thesis onderzoeken we het concept van stochastische visibiliteit. De zichtbaarheidsbepaling tussen twee punten in een driedimensionale scene is een veel voorkomende, maar kostelijke operatie. De zichtbaarheidsbepaling vereist dat we alle geometrische primitieven tussen de twee punten moeten testen op een intersectie. Met behulp van stochastische visibiliteit splitsen we de visibiliteit in meerdere termen die elk minder intersectie testen vereisen. We evalueren de visibiliteit stochastisch door voor elke schaduwstraal slechts één van deze termen te kiezen en te evalueren. We bewijzen dat deze stochastische visibiliteitsevaluatie zal convergeren naar het juiste resultaat en we zullen aantonen hoe de stochastische visibiliteit kan geïntegreerd worden met een praktisch belichtingsalgoritme.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie
Mens-machine communicatie

Promotor: Prof. dr. ir. Philip Dutré

Assessoren: Dr. ir. Thomas Heyman
Dr. ir. Ares Lagae

Begeleider: Prof. dr. ir. Philip Dutré